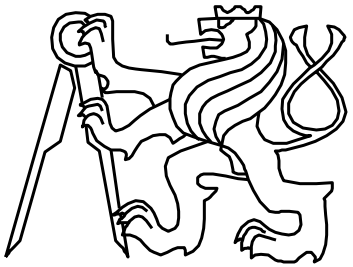
Příprava studijního programu Informatika je podporována projektem financovaným z Evropského sociálního fondu a rozpočtu hlavního města Prahy.
Praha & EU: Investujeme do vaší budoucnosti

Soubory



BI-PA1 Programování a algoritmizace 1, ZS 2012-13
Katedra teoretické informatiky
© Miroslav Balík
Fakulta informačních technologií
České vysoké učení technické

Soubory

- zápis do textového souboru
- čtení z textového souboru
- standardní textové soubory
- parametry funkce main
- zápis do binárního souboru
- čtení z binárního souboru
- přehled knihovních funkcí pro vstup a výstup
- Metoda Monte Carlo

Soubory

- Soubor je množina údajů uložená ve vnější paměti počítače, obvykle na disku
- Pro soubor jsou typické tyto operace.
    - otevření souboru
    - čtení údaje
    - zápis údaje
    - uzavření souboru
- Přístup k údajům (čtení nebo zápis) může být:
    - sekvenční (proudový)
    - libovolný, nahodilý (přímý)
- Soubory se sekvenčním přístupem (sekvenční soubory) umožňují pouze postupné (sekvenční) čtení nebo zápis údajů
- Soubory s nahodilým přístupem umožňují nahodilé čtení nebo zápis údaje (podobně jako pole)
- Způsob přístupu k údajům v souboru není zakódován v souboru, ale je dán programem a samozřejmě technickou podporou
- Budeme se zabývat pouze sekvenčními soubory

Výstup do textového souboru

Příklad: Z klávesnice přečteme řadu čísel zakončenou nulou a zapíšeme je do textového souboru, jehož jméno je dáno programem (prog9-1a.c)

```
int main(void) {
    int cislo, i = 0;
    char jmeno[] = "soubor1.txt";
    FILE *soubor;
    soubor = fopen(jmeno, "w");
    printf("zadej radu celych cisel zakoncenou nulou\n");
    do { scanf("%d", &cislo);
         fprintf(soubor, "%d ", cislo);
         i++;
         if (i==10) fprintf(soubor, "\n");
    } while (cislo);
    fclose(soubor);
    printf("cisla byla ulozena do souboru %s\n", jmenoSouboru);
    return 0;
}
```

deklarace proměnné reprezentující soubor

otevření textového souboru pro zápis

výstupní konverze, do souboru

uzavření souboru

Vstup z textového souboru

- Příklad: Z textového souboru přečteme řadu čísel zakončenou nulou, vypíšeme je na displej a pak vypíšeme jejich součet (prog9-1b.c)

```
int main(void) {
    int cislo, soucet = 0;
    char jmenoSouboru[] = "soubor1.txt";
    FILE *soubor;
    soubor = fopen(jmenoSouboru, "r");
    do {
        fscanf(soubor, "%d", &cislo);
        printf("%d ", cislo); soucet += cislo;
    } while (cislo);
    fclose(soubor);
    printf("\nsoucet cisel je : %d\n", soucet);
return 0;
}
```

otevření textového souboru pro čtení

vstupní konverze z textového souboru

uzavření textového souboru

Funkce pro otevření výstupního textového souboru

- Jméno souboru můžeme zadat z klávesnice
- Pro čtení jména souboru a otevření výstupního textového souboru zavedeme funkci (prog9-2a.c):

ošetření délky vstupu:

int res = scanf("%39s", jmenSouboru)
res == 0 … nesprávný formát
res == 1 … O.K.
res == -1 … EOF = konec vstupu

```
FILE *otevriVystupniSoubor() {
    FILE *soubor;
    char jmenoSouboru[40];
    printf("zadej jmeno vystupniho souboru: ");
    scanf("%s", jmenoSouboru);
    soubor = fopen(jmenoSouboru, "w");
    if (!soubor) {
        printf("spatne jmeno souboru\n");
        exit(1);
    }
return soubor;
}
```

Možné chyby při otvírání souboru

```
#include <errno.h>  /* obsahuje definici globální proměnné errno, */
#include <stdio.h>   /* pro  ukládání kódů chyby v případě neúspěchu */
                     /* některých standardních funkcí*/
#include <string.h> /* obsahuje deklaraci funkce strerror(), vrací řetězec */
int main (void) {       /* odpovídající kódu chyby v errno*/
    char *jmSouboru = "neco.txt";
    FILE *soubor = fopen(jmSouboru,"r");
    if (soubor!=NULL) { fclose(soubor);
        printf("Soubor %s se podarilo otevrit.\n",jmSouboru);
    } else { int chyba = errno;         /*aby nám ji nepřepsala další chyba*/
        printf("Soubor %s se nepodarilo otevrit. Chyba:%s\n",
            jmSouboru,strerror(chyba));
    };
    return 0;
}
```

Funkce pro otevření výstupního textového souboru – funkce main

```
int main(void) {
        int cislo, i = 0;
        FILE *soubor = otevriVystupniSoubor();
        printf("zadej radu celych cisel zakoncenou nulou\n");
        do {
                scanf("%d", &cislo);
                fprintf(soubor, "%d ", cislo);
                i++;
                if (i==10) fprintf(soubor, "\n");
        } while (cislo);
        fclose(soubor);
        printf("cisla byla ulozena do souboru\n");
        system("PAUSE");
        return 0;
}
```

Věděli jste, že funkce printf vrací počet znaků, které byly funkcí zapsány do stdout?

Funkce pro otevření vstupního textového souboru

```
FILE *otevriVstupniSoubor() {
        FILE *soubor;
        char jmenoSouboru[40];
        printf("zadej jmeno vstupniho souboru: ");
        scanf("%s", jmenoSouboru);
        soubor = fopen(jmenoSouboru, "r");
        if (!soubor) {
                printf("souboru nenalezen\n");
                system("PAUSE");
                exit(1);
        }
        return soubor;
}
```

Funkce pro otevření vstupního textového souboru - main

```
/*prog9-2b.c */
int main(void) {
        int cislo, soucet = 0;
        FILE *soubor = otevriVstupniSoubor();
        do {
                fscanf(soubor, "%d", &cislo);
                printf("%d ", cislo); soucet += cislo;
        } while (cislo);
        fclose(soubor);
        printf("\nsoucet cisel je : %d\n", soucet);
        system("PAUSE");
        return 0;
}
```

Ukončení čtení při dosažení konce souboru

- Pro ukončení čtení textového souboru lze využít toho, že funkce *fscanf* vrátí hodnotu *EOF,* když v souboru už nejsou žádná data (prog9-3.c)
  - je-li fukce *fscanf* volána a v souboru již nic není (bylo dosaženo konce souboru), nezmění hodnotu proměnné a vrátí *EOF*

```
int main(void) {
   int cislo, soucet = 0;
   FILE *soubor = otevriVstupniSoubor();
   while (fscanf(soubor, "%d", &cislo)!=EOF) {
            printf("%d ", cislo);
            soucet += cislo;
   }
   fclose(soubor);
   printf("\nsoucet cisel je : %d\n", soucet);
   system("PAUSE");
   return 0;
}
```

Kopie textového souboru 1

- Program, který přečte textový soubor a vytvoří jeho kopii čtením znak po znaku. Jména souborů budou přečtena z klávesnice (prog9-4a.c):

```
int main(void) {
    int znak;
    printf("vstupni soubor, ");
    FILE *vstup = otevriSoubor("r");
    printf("vystupni soubor, ");
    FILE *vystup = otevriSoubor("w");
    while (fscanf(vstup, "%c", &znak)==1)
        fprintf(vystup, "%c", znak);
    fclose(vstup);
    fclose(vystup);
    system("PAUSE");
    return 0;
}
```

zde využíváme toho, že funkce f*scanf* před dosažením konce souboru vrací počet přečtených položek

Funkce pro otevření souboru s parametrem

- Způsob (mód) otevření souboru bude dán parametrem

```
FILE *otevriSoubor(char* mode) {
        FILE *soubor;
        char jmenoSouboru[40];
        printf("zadej jmeno souboru: ");
        scanf("%s", jmenoSouboru);
        soubor = fopen(jmenoSouboru, mode);
        if (!soubor) {
                printf("spatne jmeno souboru\n");
                system("PAUSE");
                exit(1);
        }
        return soubor;
}
```

Kopie textového souboru 2

- Pro čtení znaku a výstup znaku použijeme funkce *fgetc* a *fputc* (prog9-4b.c):

```
int main(void) {
    int znak;
    printf("vstupni soubor, ");
    FILE *vstup = otevriSoubor("r");
    printf("vystupni soubor, ");
    FILE *vystup = otevriSoubor("w");
    while ((znak=fgetc(vstup))!=EOF)
        fputc(znak, vystup);
    fclose(vstup);
    fclose(vystup);
    system("PAUSE");
    return 0;
}
```

funkce *fgetc* vrátí přečtený znak, pokud je konec souboru, vrátí *EOF,* což je -1 (proto proměnná *znak* musí být typu *int*

funkce *fputc* zapíše znak do souboru

Kopie textového souboru 3

- Program, který vytvoří kopii textového souboru čtením a zápisem po řádcích (prog9-4c.c)

```
#define MAXDELKA 100
int main(void) {
    char radek[MAXDELKA];
    printf("vstupni soubor, ");
    FILE *vstup = otevriSoubor("r");
    printf("vystupni soubor, ");
    FILE *vystup = otevriSoubor("w");
    while (fgets(radek, MAXDELKA, vstup))
        fputs(radek, vystup);
    fclose(vstup);
    fclose(vystup);
    system("PAUSE");
    return 0;
}
```

přečte řádek (nanejvýš *MAXDELKA-1* znaků) a uloží do pole *radek*

zapíše *radek* do souboru *vystup*

Standardní textové soubory

- Jsou deklarovány v stdio.h a jsou to:
  - stdin vstupní, klávesnice
  - stdout výstupní, obrazovka
  - stderr výstupní, obrazovka
- Příklad: výpis textového souboru na obrazovku (prog9-5a.c):

```
#define MAXDELKA 100
int main(void) {
        char radek[MAXDELKA];
        printf("vstupni soubor, ");
        FILE *vstup = otevriSoubor("r");
        while (fgets(radek, MAXDELKA, vstup))
                fputs(radek, stdout);
        fclose(vstup);
        system("PAUSE");
        return 0;
}
```

Standardní textové soubory

- Příklad: Vytvoření textového souboru čtením z klávesnice (prog9-5b.c):

```
#define MAXDELKA 100
int main(void) {
    char radek[MAXDELKA];
    printf("vystupni soubor, ");
    FILE *vystup = otevriSoubor("w");
    while (fgets(radek, MAXDELKA, stdin))
        fputs(radek, vystup);
    fclose(vystup);
    system("PAUSE");
    return 0;
}
```

Konec souboru při čtení z klávesnice zadáme kombinací *Ctrl-Z* (MS Windows), nebo *Ctrl-D* (UNIX)

Parametry funkce main

- Hlavní funkce *main* může být napsána jako funkce se dvěma parametry, s jejichž pomocí lze převzít parametry příkazového řádku, kterým je program spuštěn (prog9-6a.c)

```
int main(int argc, char *argv[]) {
    int i;
    printf("parametry prikazoveho radku:\n");
    printf("argc = %d\n", argc);
    for (i=0; i<argc; i++)
        printf("argv[%d]: %s\n", i, argv[i]);
    system("PAUSE");
    return 0;
}
```

- *argv*[0] cesta souboru spuštěného programu
- *argv*[1] první parametr
- …
- *argv*[*argc*-1] poslední parametr

Zadání jmen souborů příkazovým řádkem

- Kopie textového souboru, soubory zadané parametry příkazového řádku

```
/*prog9-6b.c*/
#define MAXDELKA 100
int main(int argc, char *argv[]) {
    char radek[MAXDELKA];
    if (argc<3) {
        printf("malo parametru prikazoveho radku\n");
        system("PAUSE");
        return 1;
    }
    FILE *vstup = otevriSoubor(argv[1], "r");
    FILE *vystup = otevriSoubor(argv[2], "w");
    while (fgets(radek, MAXDELKA, vstup))
        fputs(radek, vystup);
    fclose(vstup);  fclose(vystup);
    system("PAUSE");    return 0;
}
```

Vytvoření binárního souboru

- Binární soubory obsahují data ve vnitřní reprezentaci
- Příklad: z klávesnice přečteme řadu celých čísel zakončenou nulou a zapíšeme je do binárního souboru (prog9-7a.c)

```
int main(void) {
    int cislo;
    printf("vystupni binarni soubor, ");
    FILE *vystup = otevriSoubor("wb");
    do {
        fscanf(stdin, "%d", &cislo);
        fwrite(&cislo, sizeof(int), 1, vystup);
    } while (cislo);
    fclose(vystup);
    system("PAUSE");
    return 0;
}
```

fwrite(*odkud, d, n, kam*)
počínaje adresou *odkud*
zapíše do binárního souboru *kam*
n položek, každá délky *d* bytů

Čtení binárního souboru

Binární soubor vytvořený předchozím příkladem vypíšeme na obrazovku (prog9-7b.c)

```
int main(void) {
    int cislo, i=0;
    printf("vstupni binarni soubor, ");
    FILE *vstup = otevriSoubor("rb");
    while (fread(&cislo, sizeof(int), 1, vstup)==1) {
        fprintf(stdout, "%d ", cislo);
        i++;
        if (i==10) {
            fprintf(stdout, "\n"); i = 0;
        }
    }
    fclose(vstup);
    system("PAUSE");
    return 0;
}
```

fread(*kam, d, n, odkud*)
z binárního souboru *odkud*
přečte *n* položek, každá délky *d* bytů,
do paměti počínaje adresou *kam*
a vrátí počet přečtených položek

Rozdíl mezi textovými a binárními soubory

- Textové soubory se člení na řádky
- Oddělovačem řádků v paměti (v řetězcích) je jediný znak `'\n'` s kódem 10 (*LF*)
- Oddělovačem řádků v souboru je:
  - v OS Windows dvojice znaků *CR LF* s kódy 13 a 10
  - v OS Unix (Linux a pod.) je jediný znak *LF*
- Rozdílné reprezentace oddělovače řádků v souboru řeší knihovna C tak, že v OS Windows:
  - při čtení textového souboru se dvojice bytů s hodnotami 13 a 10 přečte jako jediný znak s kódem 10 (*LF*)
  - při zápisu do textového souboru se znak s kódem 10 zapíše jako dvojice bytů s hodnotami 13 a 10
- Při otevírání binárních souborů, jejichž obsahem nejsou znaky a nečlení se na řádky, je třeba výše uvedenou transformaci potlačit uvedením příznaku *b*

Rozdíl mezi textovými a binárními soubory

```
/*prog9-7c.c*/
int main(void) {
    int c;
    FILE *vystup = fopen("data.bin", "w");
    for (c=5; c<15; c++)
        fputc(c, vystup);
    fclose(vystup);
    FILE *vstup = fopen("data.bin", "rb");
    while ((c=fgetc(vstup))!=EOF)
        printf("%d ", c);
    printf("\n");
    fclose(vstup);
    system("PAUSE");  return 0;
}
```

Program vypíše 5 6 7 8 9 13 10 11 12 13 14

Shrnutí vstupu a výstupu

- Standardní knihovna <stdio.h>
- Vstup a výstup:
  - textový (formátovaný)
  - binární (neformátovaný)
- Textový vstup a výstup
  - standardní: printf, putchar, scanf, getchar
  - z/do souboru: fprintf, putc, fscanf, getc
  - z/do paměti: sprintf, sscanf
  - *přímo na obrazovku: cprintf*
- Standardní textové soubory:
  - stdin vstupní, klávesnice, lze přesměrovat
  - stdout výstupní, obrazovka, lze přesměrovat
  - stderr výstupní, obrazovka

Standardní výstup

- **int putchar(int c);**
  - výstup znaku c;
  - při chybě vrací EOF (-1), jinak c
- **int puts(char *s);**
  - výstup řetězce s + nový řádek
- **int printf(char *format, ...);**
  - výstup hodnot různých typů;
  - vrací počet vytisknutých znaků
  - formát:

    řetězec - popisuje typ a tvar argumentů

    výstupní konverze: začíná znakem %

    ostatní znaky kopírovány na výstup
  - konverze:

    % [*příznaky*] [*šířka*] [. *přesnost*] [*velikost*] *typ*

Výstupní konverze

příznaky

-	zarovnání doleva (jinak doprava)
+	u číselných typů se znaménkem vždy výstup znamenka
mezera	místo znaménka plus výstup mezery
#	alternativní výstup (viz dále)
0	výplňovým znakem je nula (v kombinaci s – je ignorován)

šířka

číslo	minimální celková šířka (včetně znamének, exponentu, ...) výplňovým znakem je implicitně mezera implicitní způsob zarovnání je doprava
*	šířka je dána parametrem typu int

Výstupní konverze (pokračování)

přesnost

číslo	pro celočíselné konverze minimální počet číslic pro reálné konverze počet desetinných míst pro řetězce maximální počet znaků

velikost

h	pro celočíselné typy: parametr typu (unsigned) short int
l	pro celočíselné typy: parametr typu (unsigned) long int
L	pro reálné typy: parametr typu long double

Výstupní konverze (pokračování)

typ	
d, i	int vypisován dekadicky
o	unsigned int oktalově (s příznakem # i prefix 0)
u	unsigned int dekadicky
x, X	unsigned int hexadecimálně: písmena abcdef resp ABCDEF (s příznakem # i prefix 0x resp. 0X)
f	double ve tvaru 99.999 (implicitní hodnota přesnosti je 6)
e, E	double ve tvaru 9.9999e±99 před desetinnou tečkou vždy jedna nenulová číslice implicitní hodnota přesnosti je 6
g, G	double ve tvaru konverze f nebo e resp.E přesnost udává počet platných číslic
c	int konvertován na unsigned char a vypsán jako znak
s	řetězec (char*) vypsán po znacích
p	ukazatel (void*) vypsán ve tvaru závislém na implementaci
n	parametr typu int*: celočíselná proměnná je naplněna počtem dosud vypsaných znaků
%	vypíše se %; celá konverze vypadá %%

Příklady výstupní konverze

Příklady (místo mezery je použito podtržítko)

konverze	hodnota	výstup
%d	-300	-300
%-5i	-5	-5___
%5x	0xABC	__abc
%#09X	0x12AB	0X00012AB
%6.2f	4.352	__4.35
%+6.0f	100.44	__+100
%10.3E	100.44	_1.004E+02
%*.*f	7, 2, 100.123	_100.12
%c	'a'	a
%4s	"hello"	hello
%.4s	"hello"	hell
%-10.4s	"hello"	hello_____

Standardní vstup

- **int getchar(void);**
  - vstup znaku (při chybě, konci souboru EOF)
- **char* gets(char *s);**
  - vstup řetězce do konce řádku (nebezpečí: není omezena délka řetězce)
- **int scanf(char *format, ...);**
  - vstup hodnot různých typů; argumenty musejí být ukazatelé
  - vrací počet úspěšně zkonvertovaných a přiřazených parametrů (s výjimkou koverze n)
  - formát

    platí obdobná pravidla jako u printf

    konverze uvedeny znakem %

    mezery ignorovány (s výjimkou konverze c, [a n)

    ostatní znaky srovnávány se vstupem
  - konverze:

    % [*] [šířka] [velikost] *typ*

Vstupní konverze

* položka je konvertována, ale není přiřazena

Šířka

- číslo maximální celková šířka (bez úvodních mezer)

Velikost

- viz velikost u printf
- l u konverze f, e, E, g, G: parametr typu double * (jinak float *)

Typ

d	dekadické celé číslo; parametr typu int *
i	celé číslo v libovolné soustavě (8, 10, 16); parametr typu int *
o	oktalové celé číslo; parametr typu unsigned int *
x, X	šestnáctkové celé číslo; parametr typu unsigned int *
f, e, E, g, G	reálné číslo; parametr typu float * !!!
s	posloupnost nemezerových znaků; parametr typu char *; řetězec musí být dostatečně dlouhý!
c	posloupnost znaků (implicitní šířka je 1); parametr typu char *
[	posloupnost

Formátovaný vstup a výstup v paměti

- int sscanf(char *Buffer, char *Format, ...);
  - vstup z řetězce Buffer
- int sprintf(char *Buffer, char *Format, ...);
  - výstup do řetězce Buffer
- Příklady:
  - sscanf(Retezec, "%s = %f", &S, &F);
    - » Retezec == "Výsledek = 4.52"
    - » S == "Výsledek"
    - » F == 4.52
  - sprintf(Jmeno, "Temp%d", I);
    - » I == 3
    - » Jmeno == "Temp3"

Otevření a zavření souboru

- **FILE *fopen(char *Jmeno, char *Mod);**
  - otevření souboru specifikovaného Jména
  - Jmeno - může obsahovat cestu k souboru (pozor: \\)
  - Mod - způsob otevření souboru; řetězec složen ze znaků:

r	otevření na začátku (čtení)
w	existující obsah vymazán (psaní)
a	otevření na konci (zápis na konec)
b	binární soubor (jinak textový)
+	update

  - povolené kombinace:

"r", '', "a"	textový soubor
"rb", "wb", "ab"	binární soubor
"r+", "w+", "a+"	textový soubor, update
"rb+" = "r+b", "wb+" = "w+b", "ab+" = "a+b"	binární soubor, update

- **int fclose(FILE *Soubor);**
  - zavření souboru

Textový vstup a výstup

```
int fgetc(FILE *Soubor);
int getc(FILE *Soubor);
char *fgets(char *Retezec, size_t Max, FILE *Soubor);
```

čte nejvýše Max znaků; případný konec řádku je součástí řetězce!

```
int fscanf(FILE *Soubor, char *Format, ...);
int fputc(int Znak, FILE *Soubor);
int putc(int Znak, FILE *Soubor);
int fputs(char *Retez, FILE *Soubor);
```

nepřidává přechod na nový řádek

```
int fprintf(FILE *Soubor, char *Format, ...);
```

Binární vstup a výstup

- **size_t fread(void *Kam, size_t Delka, size_t Pocet, FILE *Soubor);**
  - vrací počet skutečně přečtených položek
- **size_t fwrite(void *Odkud, size_t Delka, size_t Pocet, FILE *Soubor);**
  - vrací počet skutečně zapsaných položek

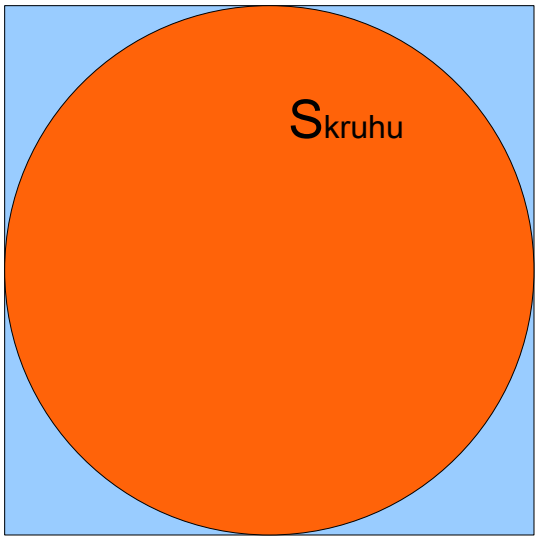
Přímý přístup

- **int fseek(FILE *Soubor, long Offset, int Odkud);**
  - nastaví pozici v Souboru
  - Odkud může nabývat hodnot:
    - SEEK_SET (od začátku souboru)
    - SEEK_CUR (od aktuální pozice)
    - SEEK_END (od konce souboru)
- **long ftell(FILE *Soubor);**
  - vrací pozici v souboru

Metoda Monte Carlo

- třída algoritmů pro simulaci systémů
- stochastické metody používající pseudonáhodná čísla
- typicky využívány pro výpočet (vícerozměrných)integrálů, řešení diferenciálních rovnic
- Základní myšlenka: chceme určit střední hodnotu veličiny, která je výsledkem náhodného děje. Vytvoří se počítačový model toho děje a po proběhnutí dostatečného množství simulací se mohou data zpracovat klasickými statistickými metodami, třeba určit průměr a směrodatnou odchylku.

Metoda Monte Carlo – výpočet čísla Π

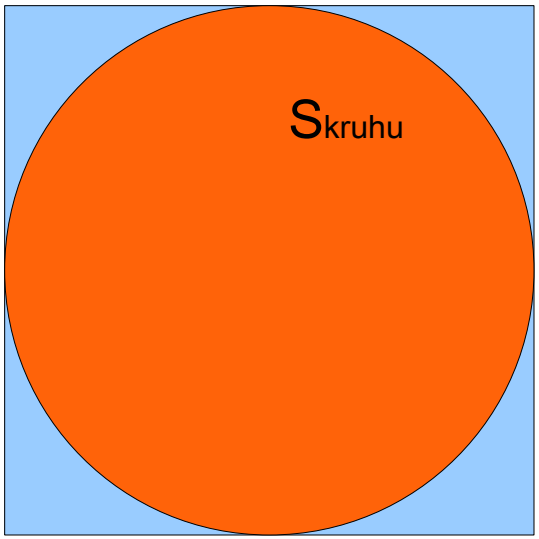


Nechť je poloměr kruhu r, pak
$S_{kruhu} = Π*r*r/4$
$S_{celku} = r*r$

$Π = 4 * S_{kruhu} / S_{celku}$

Algoritmus:
1. opakovaně budeme generovat body padající do čtverce
2. do proměnné in budeme počítat ty, které padnou dovnitř
3. do proměnné all počet pokusů
4. Π = 4 * in/all

Metoda Monte Carlo – výpočet čísla Π




```
/* montecarlo.c */
```

Přesnost algoritmu ovlivní:
1. počet opakování
2. kvalita generátoru pseudonáhodných čísel
3. zaokrouhlovací chyby