Standardní proudy, soubory a datové struktury

IZP-cv05

Ing. Jakub Husa

Vysoké Učení Technické v Brně, Fakulta informačních technologií
Božetěchova 1/2. 612 66 Brno - Královo Pole
ihusa@fit.vut.cz

26. října 2023

Standardní proudy

Standardní proudy

Základním komunikačním rozhraním programu jsou datové proudy:

- Knihovna stdio.h poskytuje tři standardní proudy (stdin, stdout, stderr) .
- stdin – standardní vstup, ze kterého čte funkce scanf (obvykle klávesnice).
- stdout – standardní výstup, do kterého zapisuje funkce printf (obvykle konzole).
- stderr – standardní chybový výstup (obvykle konzole).

Při spuštění z příkazové řádky standardní proudy můžeme přesměrovat na nějaký soubor (nebo program) směrovacími značkami (<, >, >>, 2>, |):

- `./main < input.txt` – vstup z klávesnice nahraď souborem input.txt.
- `./main > output.txt` – výstup do konzole nahraď souborem output.txt.
- `./main >> output.txt` – výstup přidej na konec souboru output.txt.
- `./main 2> error.txt` – chybový výstup přesměruj do souboru error.txt.
- `./main | ./main2` – výstup programu main přesměruj na vstup main2.

Směrovací značka ani název souboru **NEJSOU** součástí argumentů programu:

- Vývojové prostředí Code::Blocks přesměrování proudů nepodporuje.

Návratovou hodnotou funkce scanf je počet úspěšně načtených položek:

- Návratová hodnota umožňuje ověřit úspěšnost načtení vstupu.

```
1  int x, y;                //dve cela cisla "x" a "y"
2  y = scanf("%i", &x);     //do "x" nacteme vstup
3                           //a do "y" priradime navratovou hodnotu
4  if (y > 0)               //pokud se cteni podarilo
5      printf("Cteni se podarilo (x = %i)\n", x);
6  else                     //jinak (pokud cteni selhalo)
7      printf("Cteni selhalo\n");
```

Pokud scanf narazí na konec souboru, vrátí konstantu EOF (end-of-file):

- Na klávesnici EOF zadáme zkratkou ctrl+z (Windows), nebo ctrl+d (Unix).
- Umožňuje nám načítat celý vstupní soubor.

```
8  char y[101];             //vytvarime si pole znaku
9  while (scanf("%100[^\n]\n", y) != EOF) //dokud nenarazime na konec souboru
10 {                        //nacitame retezec (i s mezerami) a konec radku
11     printf("%s\n", y); //vypisujeme nacteny retezec
12 }
```

Na procvičení

Vyzkoušejte si:

- Napište program který bude ze vstupu načítat celá čísla tak dlouho dokud je to možné, a spočítá jejich sumu.
- Na serveru merlin.fit.vutbr.cz si vytvořte soubory cisla.txt a numbers.txt, naplňte je vzorovými daty, a přesměrujte je na vstup vašeho programu.

Soubor cisla.txt:

```
1 10 20
```

Soubor numbers.txt:

```
1 10 20 30 40 X 50
```

Například:

- `./main < cisla.txt` => Suma je 30
- `./main < numbers.txt` => suma je 100

Na procvičení

Vyzkoušejte si:

- Vytvořte si dvě pole do kterých se vejdou řetězce o velikosti až 100 znaků.
- Napište program který bude ze vstupu načítat dvojice řádků, a jejich obsah (oddělený čárkou) vypisovat na výstup dokud nenarazí na konec souboru.
- Přihlaste se na server merlin.fit.vutbr.cz a program na něm přeložte.
- Na serveru si vytvořte soubor input.txt, naplňte ho vzorovými daty, a přesměrujte ho na standardní vstup vašeho programu.

Vstup (input.txt):

```
1 Petr Dvorak
2 603123456
3 Jana Novotna
4 777987654
5 Bedrich Smetana ml.
6 541141120
```

Výstup (stdout):

```
1 Petr Dvorak, 603123456
2 Jana Novotna, 777987654
3 Bedrich Smetana ml., 541141120
```

Soubory

Otevření a uzavření souboru

Soubor otevíráme funkcí fopen z knihovny stdio.h:

- Vstupem funkce jsou název souboru a režim otevření (`"r"` – read, `"w"` – write).
- Výstupem je proud s datovým typem – `FILE*` – (adresa otevřeného souboru).
- Pokud se soubor nepodařilo otevřít, funkce vrátí chybovou adresu NULL.

Úspěšně otevřený soubory uzavíráme funkcí fclose:

- Jejím vstupem je proud který chceme uzavřít.

```
1  FILE* vstup;                        //promenna typu "proud" jmenem "vstup"
2  vstup = fopen("input.txt", "r"); //soubor "input.txt" otevreme pro "cteni"
3
4  if (vstup != NULL)                  //pokud se otevreni podarilo
5  {
6      printf("Otevreni se podarilo\n"); //vypis
7      fclose(vstup);                  //zavirame soubor
8  }
9  else                                //jinak (otevreni selhalo)
10 {
11     printf("Otevreni selhalo\n");//vypis
12 }
```

Čtení a zápis souboru

Soubory čteme a zapisujeme funkcemi fscanf a fprintf z knihovny stdio.h:

- Funkce se požívají stejně jako scanf a printf, ale jako první parametr jim předáváme proud ze kterého mají číst nebo do kterého mají zapisovat.

```
1  FILE* vstup;                              //proud jmenem "vstup"
2  FILE* vystup;                             //proud jmenem "vystup"
3
4  vstup = fopen("input.txt", "r");          //"input.txt" otevirame pro cteni
5  if (vstup != NULL)                        //pokud se otevreni podarilo
6  {
7      vystup = fopen("output.txt", "w"); //"output.txt" otevirame pro zapis
8      if (vystup != NULL)                   //pokud se otevreni podarilo
9      {
10         char x[101];                      //pole znaku
11         fscanf(vstup, "%100s", x);        //z "input.txt" nacitame retezec
12         fprintf(vystup, "%s\n", x);       //a zapisujeme ho do "output.txt"
13
14         fclose(vystup);                   //zavirame soubor "output.txt"
15     }
16     fclose(vstup);                        //zavirame soubor "input.txt"
17 }
```

Chybové hlášení

Pokud program končí s chybou, měl by vypisovat nějaké chybové hlášení:

- Chybová hlášení zapisujeme do proudu stderr (standardní chybový výstup).

```
1  FILE* vstup;                                //"proud" jmenem "vstup"
2  vstup = fopen ("input.txt", "r");           //"input.txt" otevirame pro "cteni"
3
4  if (vstup != NULL)                          //pokud se otevreni podarilo
5  {
6      printf("Otevreni se podarilo\n");       //vypis
7      fclose(vstup);                          //zavirame soubor
8      return 0;                               //program konci bez chyby
9  }
10 else                                        //jinak (otevreni selhalo)
11 {
12     fprintf(stderr, "Chyba: soubor se nepodarilo otevrit\n");
13                                             //vypis chyboveho hlaseni
14     return 1;                               //program konci s chybou
15 }
```

Čtení nebo zápis adresy NULL způsobí neplatný přístup do paměti:

- Neuzavření souboru otevřeného v režimu pro zápis ho **POŠKODÍ**!

Na procvičení

Vyzkoušejte si:

- Vytvořte si soubory cisla.txt a znaky.txt a naplňte je vzorovými daty.
- Napište program který jako argument dostane název souboru.
- Soubor otevřete v režimu pro čtení a spočítejte kolik obsahuje znaků.
- Pokud došlo k chybě vypište chybové hlášení a program ukončete s chybovou návratovou hodnotou.

Soubor cisla.txt:

```
1 10 20
```

Soubor znaky.txt:

```
1 aaa
2 bbb
3 ccc
```

Například:

- `./main cisla.txt` => Soubor cisla.txt obsahuje 5 znaku => 0
- `./main znaky.txt` => Soubor znaky.txt obsahuje 11 znaku => 0
- `./main err` => Soubor err se nepodarilo otevrit => 1
- `./main` => Nedostatek argumentu => 2

Datové struktury

Definice struktury

Struktury umožňují spojení položek libovolného datového typu:

- Struktury usnadňují organizaci dat a předávání parametrů funkcím.
- Definice obsahuje klíčové slovo – `struct` –, identifikátor a tělo.
- Tělo struktury obsahuje seznam položek a píšeme za ním středník (;).

```
1  struct Sdvojice     //definujeme strukturu jmenem "Sdvojice"
2  {                   //se dvema polozkami
3      int cislo;      //polozka typu "int"  jmenem "cislo"
4      char znak;      //polozka typu "char" jmenem "znak"
5  };                  //POZOR -- za telem struktury piseme strednik!
```

Při vytváření proměnné opakujeme klíčové slovo – `struct` –:

- K položkám struktury přistupujeme tečkou (.) a identifikátorem:

```
6  struct Sdvojice A; //promenna typu "struktura Sdvojice" jmenem "A"
7  A.cislo = 10;      //jeji polozku "cislo" nastavujeme na "10"
8  A.znak = 'X';      //jeji polozku "znak"  nastavujeme na "X"
```

Struktury načítáme a vypisujeme po jednotlivých položkách:

- Počáteční hodnotu položek můžeme inicializovat složenými závorkami ({}).

```
1  struct Sdvojice     //definujeme strukturu jmenem "Sdvojice"
2  {                   //se dvema polozkami
3      int cislo;      //polozka typu "int"  jmenem "cislo"
4      char znak;      //polozka typu "char" jmenem "znak"
5  };                  //POZOR -- za telem struktury piseme strednik!
6
7  int main()                          //zacatek programu
8  {
9      struct Sdvojice A = {10, 'X'}; //"Sdvojice" jmenem "A" s polozkami
10                                     //"A.cislo = 10" a "A.znak = X"
11     printf("Cislo je %i, znak je %c\n", A.cislo, A.znak);
12
13     scanf("%i", &A.cislo);          //nacitame polozku "cislo" promenne "A"
14     scanf("%c", &A.znak);           //nacitame polozku "znak"  promenne "A"
15     printf("Cislo je %i, znak je %c\n", A.cislo, A.znak);
16     return 0;                       //konec programu
17 }
```

Deklarace datového typu

Klíčovým slovem – `typedef` – pro strukturu můžeme deklarovat nový datový typ:

```
1  typedef struct Sdvojice dvojice; //deklarujeme novy datovy typ pro
2                                   //"struct Sdvojice" jmenem "dvojice"
3  struct Sdvojice   //definujeme strukturu jmenem "Sdvojice"
4  {                 //se dvema polozkami
5      int cislo;    //prvni polozka
6      char znak;    //druha polozka
7  };
```

Nový datový typ můžeme deklarovat už při definici struktury:

```
8  typedef struct Sdvojice   //deklarujeme datovy typ pro
9  {                         //"struct Sdvojice" se dvema polozkami
10     int cislo;    //prvni polozka
11     char znak;    //druha polozka
12 } dvojice;        //jmeno tohoto typu je "dvojice"
```

Při vytváření proměnné už klíčová slova nejsou potřeba:

```
13 dvojice A;         //promenna typu "dvojice" jmenem "A"
14 A.cislo = 10;      //jeji polozku "cislo" nastavujeme na "10"
15 A.znak = 'X';      //jeji polozku "znak"  nastavujeme na "X"
```

Deklarace a definice struktur

Struktury můžeme používat jako parametry funkcí i jejich návratový typ:

- Zdrojový kód programu píšeme v tomto pořadí:
  Knihovny -> nové typy -> struktury -> deklarace funkcí -> definice funkcí.

```
1  dvojice inkrementuj (dvojice A); //deklarace funkce "inkrementuj"
2
3  int main()                       //zacatek programu
4  {
5      dvojice A = {10, 'X'};       //promenna "A" s hodnotami "10" a "X"
6      A = inkrementuj(A);          //volani funkce "inkrementuj"
7      printf("Znak je %c, cislo je %i\n", A.znak, A.cislo); //vypis
8      return 0;                    //konec programu
9  }
10
11 dvojice inkrementuj (dvojice A) //definice funkce "inkrementuj"
12 {
13     dvojice B;                   //vytvarime novou promennou "B"
14     B.cislo = A.cislo + 1;       //pocitame prvni  polozku
15     B.znak = A.znak + 1;         //pocitame druhou polozku
16     return B;                    //vracime promennou "B"
17 }
```

Vyzkoušejte si:

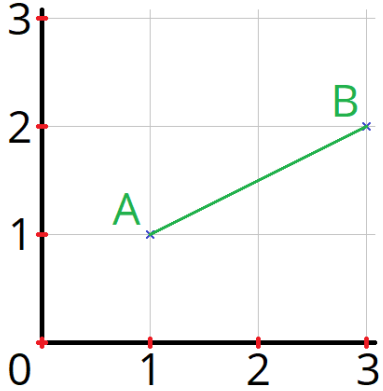
- Deklarujte datový typ bod pro strukturu obsahující dvě desetinná čísla (X a Y), představující souřadnice bodu ve dvourozměrném prostoru.
- Napište funkci vzdalenost která spočítá vzdálenost mezi dvěma body.
- Ve funkci main si vytvořte dva body, načtěte do nich nějaké souřadnice, a vypište jejich vzdálenost.

- Funkce vzdalenost je deklarována tímto způsobem:

```
1  float vzdalenost(bod A, bod B);
```

Například:

- ((0.0, 0.0), (0.0, 0.0)) => Vzdalenost bodu je 0.000
- ((1.0, 1.0), (3.0, 2.0)) => Vzdalenost bodu je 2.236
- ((0.0, 3.0), (4.0, 0.0)) => Vzdalenost bodu je 5.000



Struktury můžeme přiřazovat pouze pokud mají stejný datový typ:

- Pozor – shodný počet, pořadí a typ položek pro přiřazení nestačí!

```
1  typedef struct Sxxx //deklarujeme datorvy typ pro strukturu
2  {                   //jmenem "Sxxx" s jednou polozkou
3      int cislo;      //prvni polozka
4  } xxx;              //jmeno tohoto typu je "xxx"
5
6  typedef struct Syyy //deklarujeme datorvy typ pro strukturu
7  {                   //jmenem "Syyy" s jednou polozkou
8      int cislo;      //prvni polozka
9  } yyy;              //jmeno tohoto typu je "yyy"
10
11 int main()
12 {
13     xxx A = {10};  //promenna "A" s hodnotou "A.cislo = 10"
14     xxx B = A;     //do promenne "B" prirazujeme promennou "A"
15     yyy C = A;     //CHYBA - do promenne "C" nelze priradit promennou
16     return 0;      //       - "A" protoze nejsou stejneho datoveho typu
17 }
```

Zanořené struktury

Položkou struktury může být i další struktura nebo pole:

- K položkám zanořené struktury přistupujme pomocí několika teček (.).

```
1  typedef struct Susecka    //deklarujeme datovy typ pro strukturu
2  {                         //jmenem "Susecka" se dvema polozkami
3      bod A;                //prvni je typu "bod" a jmenuje se "A"
4      bod B;                //druha je typu "bod" a jmenuje se "B"
5  } usecka;                 //jmeno tohoto typu je "usecka"
6
7  int main()
8  {
9      usecka U;             //vytvarime usecku jmenem "U"
10     scanf("%f", &U.A.x); //nacti desetinne cislo "x" bodu "A" usecky "U"
11     scanf("%f", &U.A.y); //nacti desetinne cislo "y" bodu "A" usecky "U"
12     scanf("%f", &U.B.x); //nacti desetinne cislo "x" bodu "B" usecky "U"
13     scanf("%f", &U.B.y); //nacti desetinne cislo "y" bodu "B" usecky "U"
14
15     printf("Usecka zacina v bode (%.1f, %.1f)\n", U.A.x, U.A.y); //vypis
16     printf("Usecka konci  v bode (%.1f, %.1f)\n", U.B.x, U.B.y); //vypis
17     printf("Delka usecky je %f\n", vzdalenost(U.A, U.B));        //vypis
18     return 0;
19 }
```

Na procvičení

Vyzkoušejte si:

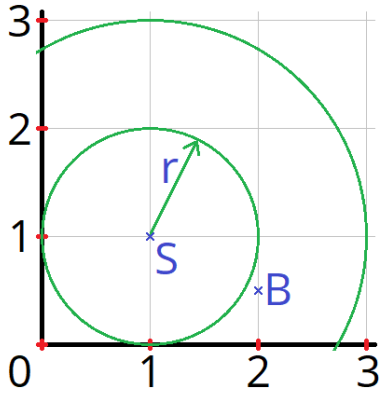
- Deklarujte datový typ kruznice pro strukturu obsahující jeden bod (S) a jedno desetinné číslo (r), představující střed a poloměr kružnice.
- Napište funkci jeUvnitr která ověří jestli se bod nachází uvnitř kružnice.
- Ve funkci main si vytvořte kružnici a bod, načtěte do nich nějaké souřadnice, a vypište jestli bod v kružnici je nebo není.

- Funkce jeUvnitr je deklarována tímto způsobem:

```
1  bool jeUvnitr(kruznice K, bod B);
```

Například:

- (((1.0, 1.0), 2.0), (2.0, 0.5)) => Bod je v kruznici
- (((1.0, 1.0), 1.0), (2.0, 0.5)) => Bod neni v kruznici



Na procvičení

Vyzkoušejte si:

- Deklarujte datový typ trojuhelnik obsahující tři body, představující jeho vrcholy.
- Napište funkce které spočítají obvod, obsah a teziste daného trojúhelníku.
- Ve funkci main si vytvořte trojúhelník, načtěte souřadnice jeho vrcholů, a vypište výsledek jednotlivých funkcí.

- Funkce jsou deklarovány tímto způsobem:

```
1  float obvod(trojuhelnik T);
2  float obsah(trojuhelnik T);
3  bod teziste(trojuhelnik T);
```

Například:

- ((0.0, 0.0), (1.0, 0.0), (0.0, 1.0)) =>
  => Obvod = 3.414
  => Obsah = 0.500
  => Teziste = (0.333, 0.333)
- ((0.0, 0.0), (2.0, 0.0), (1.0, 2.0)) =>
  => Obvod = 6.472
  => Obsah = 2.000
  => Teziste = (1.000, 0.667)