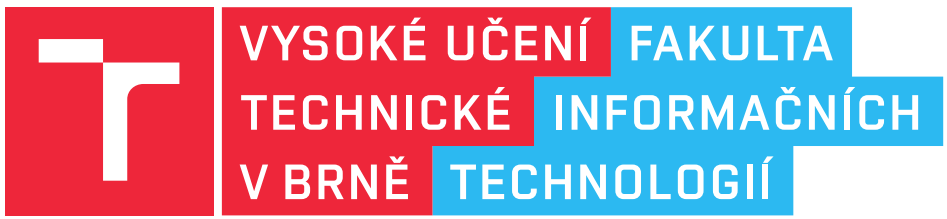
Základy programovania
Cvičenie 5

Dátové štruktúry

Cvičiaci: Ing. Magdaléna Ondrušková, (iondruskova)



15. října 2024

Dátové prúdy

- Základný mechanizmus pre prenos údajov medzi programom a napr. vonkajšími súbormi
- Funkcie podporujúce toto správanie sa nachádzajú v knižnici `stdio.h`

Tri štandardné prúdy:

- **stdin** - štandardný vstup, používa sa na čítanie vstupu, obvykle z klávesnice (napr. použitím funkcie `scanf`)
- **stdout** - štandardný výstup, používa sa na zápis výstupu, obvykle do konzoly (napr. použitím funkcie `printf`)
- **stderr** - štandardný chybový výstup, používa sa na zápis chybových výstupov

Smerovacie značky

- pri spustení z príkazového riadku vieme presmerovať štandardné prúdy na nejaký súbor (program) pomocou smerovacích značiek.

```
./main < input.txt  // vstup z klavesnice nahrad suborom input.txt.
./main > output.txt // vystup do konzole nahrad suborom output.txt.
./main >> output.txt // vystup pridaj na konec suboru output.txt.
./main 2> error.txt // chybovy vystup presmeruj do souboru error.txt.
./main | ./main2   // vystup programu main presmeruj na vstup main2
```

Návratová hodnota

- Scanf vracia počet úspešne načítaných vstupov
- Umožňuje overiť úspešnosť načítania vstupov

```c
#include <stdio.h>

int main() {
    int age;
    float height;
    char initial;
    printf("Enter your age, heigh, and first initial: ");
    int result = scanf("%d %f %c", &age, &height, &initial);

    if (result == 3) {
        printf("Entered: Age = %d, Height = %.2f, Initial = %c\n",
      age, height, initial);
    } else {
        printf("Failed to read all values.\n");
    }

    return 0;
}
```

Návratová hodnota

- Vráti špeciálnu hodnotu `EOF` - end-of-file
- Znamená to, že narazil na koniec súboru
- Pomocou klávesnice: `CTRL+Z` (Windows) alebo `CTRL+D` (Linux)
- Vieme načítavať celý súbor bez toho, aby sme dopredu vedeli ako je tento súbor veľký (resp. koľko položiek obsahuje)

```
1  #include <stdio.h>
2
3  int main() {
4     char y[101]; //vytvarime si pole znaku
5      while (scanf("%100[^\n]\n", y) != EOF) //dokud nenarazime na
        konec souboru
6      {
7          printf("%s\n", y); //vypisujeme nacteny retezec
8      }
9     return 0;
10 }
```

Príkazom `scanf("%100[^\n]\n", y)` načítavame vstup aj s medzerami

Príklad - načítanie vstupu zo súboru

- Vytvorte si súbor `cisla.txt`, ktory obsahuje 5 roznych cisiel. Postupne cisla nacitajte a vypiste ich sucet.

Priklad vstupu a vystupu:

```
// vstup cisla.txt
10 20 30 40 50

// vystup:
150
```

Príklad - načítanie vstupu zo súboru

- Vytvorte si súbor `znaky.txt`, ktory obsahuje nahodne znaky. Spocitajte, kolkokrat sa prvy znak vyskytuje v subore (okrem 1. vyskytu).

Priklad vstupu a vystupu:

```
1 // vstup cisla.txt
2 a a a a b b a a b c
3
4 // pocitam teda pocet 'a' v zvysnom subore
5
6 // vystup:
7 5
```

Dátový typ `FILE`

- Slúži na reprezentáciu súborového `prúdu`

```
FILE *file_pointer;
```

Otváranie súboru pomocou funkcie `fopen`

- `filename` - názov súboru
- `mode` - mód (režim) práce so súborom

Rôzne módy práce so súbormi:

- **r** - read, otvorí súbor iba pre čítanie
- **w** - write, otvorí súbor na zápis. Ak súbor existuje, vymaže jeho obsah. Ak neexistuje, súbor sa vyzvorí.
- **a** - append, otvorí súbor na zápis. Zapisuje na koniec súboru (nezmaže predchádzajúci obsah súboru).

```
FILE *fopen(const char *filename, const char *mode);
```

Návratová hodnota funkcie `fopen`

- Ak sa súbor nepodarilo otvoriť, funkcia `fopen` vráti hodnotu `NULL`

Zatváranie súboru pomocou funkcie `fclose`

- Po skončení práce so súborom je dôležité súbor zavrieť
- Zatvorenie súboru uvoľňuje zdroje a zabezpečuje, že všetky písania sú zapísané do súboru.
- Nezavretie súboru môže viesť k poškodeniu súboru, ak je otvorený v režime zápisu.

```
1 int fclose(FILE *stream);
```

Príklad práce so súborom

```c
#include <stdio.h>

int main() {
    FILE *file = fopen("data.txt", "r"); // Otvorenie suboru na citanie
    if (file == NULL) {
        printf("Nepodarilo sa otvorit subor.\n");
        return 1; // Chyba pri otvarani
    }

    // Dalsie operacie so suborom...

    fclose(file); // Zatvorenie suboru
    return 0;
}
```

Čítanie zo súboru

- `fscanf()` - Číta formátované údaje zo súboru.
- `fgets()` - Číta jeden riadok textu zo súboru.
- `fgetc()` - Číta jeden znak zo súboru.

Príklad použitia fgets():

```c
#include <stdio.h>

int main() {
    FILE *file = fopen("data.txt", "r");
    char buffer[100];

    if (file != NULL) {
        while (fgets(buffer, sizeof(buffer), file) != NULL) {
            printf("%s", buffer); // Vypise riadok zo suboru
        }
        fclose(file);
    }
    return 0;
}
```

Zápis do súboru

- `fprintf()` - Zapisuje formátované údaje do súboru.
- `fputs()` - Zapisuje reťazec do súboru.
- `fputc()` - Zapisuje jeden znak do súboru.

Príklad použitia `fprintf`

```c
#include <stdio.h>

int main() {
    FILE *file = fopen("output.txt", "w");

    if (file != NULL) {
        // Zapisuje text do suboru
        fprintf(file, "Toto je testovaci text.\n");
        fclose(file);
    }
    return 0;
}
```

Úloha: Napíš program, ktorý načíta text zo súboru `input.txt`, prevedie všetky znaky na veľké písmená a uloží výsledok do nového súboru `output.txt`.

```
// Priklad vstupneho suboru input.txt:
a
ahoj
ahoj
a
b
Dnes je pekny den.
8.10.2024

// Priklad vystupneho suboru output.txt:
A
AHOJ
AHOJ
A
B
DNES JE PEKNY DEN.
8.10.2024
```

Definícia štrukútry

- Umožňujú spojenie položiek ľubovoľného dátového typu
- Usnadňujú organizáciu dát a predávanie parametrov do funkcie

Definujeme ju pomocou kľúčového slova `struct`.
Polozky oddelene pomocou bodkočiarky (stredníka).

```
1 struct nazov {
2     typ1 premenna1;
3     typ2 premenna2;
4     // ... dalsie premenne
5 };
```

Definovanie konkrétnej štruktúry pre osobu:

```
struct Osoba {
    char meno[50];
    int vek;
    float vyska;
};
```

Po definovaní štruktúry môžeme vytvoriť premenné tohto dátového typu:

```
struct Osoba osoba1; // Deklaracia premennej typu Osoba
```

Príp. pomocou `typedef` pre zjednodušenie:

```
typedef struct Person { // definujeme strukturu menom Person
    char meno[50];     // polozka meno
    int vek;           // polozka vek
    float vyska;       // polozka vyska
} osoba; // deklarujeme jej datovy typ "osoba"

osoba Anna; // osoba "Anna" bez pociatocnych hodnot
```

Prístup k jednotlivým položkám:

```c
#include <stdio.h>

typedef struct Person{
    int age;
    int height;
} osoba;

int main()
{
    osoba person1;
    person1.height = 167;

    printf("Vyska osoby 1: %d\n", person1.height);
    return 0;
}
```

Štruktúry môžeme použiť ako parametre funkcie aj ako ich návratový typ:

```
1  #include <stdio.h>
2
3  typedef struct Person{
4      int age;
5      int height;
6  } osoba;
7
8  void vypisOsobu(osoba o) {
9      printf("Vek: %d, Vyska: %d cm\n", o.age, o.height);
10 }
11
12
13 int main()
14 {
15     osoba person1;
16     person1.age = 25;
17     person1.height = 167;
18
19     vypisOsobu(person1);
20     return 0;
21 }
```

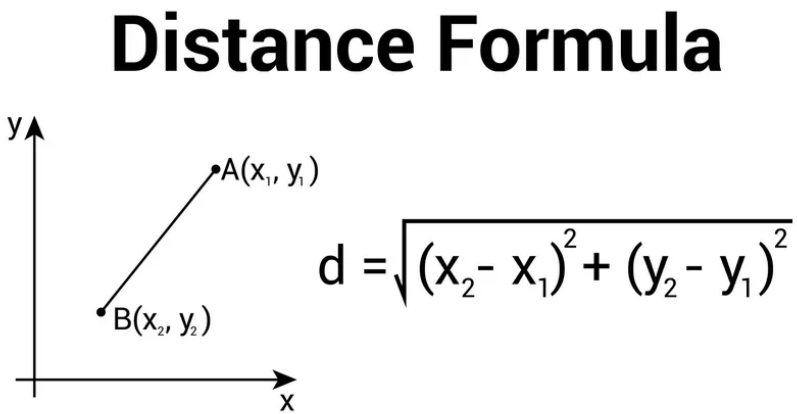
Úlohy na precvičenie

Definujte štruktúru, ktorá bude obsahovať informácie o kruhu:

- polomer
- stredová súradnica X
- stredová súradnica Y

Následne definujte funkcie, ktoré spočítajú:

- obvod kruhu `2*PI*polomer`
- obsah kruhu `PI*polomer*polomer`
- vzdialenosť medzi dvoma kruhmi (ich stredmi)



Obrázek: Príklad výpočtu vzdialenosti medzi dvoma bodmi

Položkou štruktúry môže byť aj ďalšia štruktúra:

```
1  #include <stdio.h>
2  struct Datum {
3      int den;
4      int mesiac;
5      int rok;
6  };
7  struct Osoba {
8      int vek;
9      int vyska;
10     struct Datum datumNarodenia;
11 };
12 int main(){
13     struct Osoba osoba1;
14     osoba1.vek = 25;
15     osoba1.vyska = 1.80;
16     osoba1.datumNarodenia.den = 15;
17     osoba1.datumNarodenia.mesiac = 10;
18     osoba1.datumNarodenia.rok = 1995;
19     printf("Vek: %i, Datum narodenia: %i.%i.%i.", osoba1.vek,
       osoba1.datumNarodenia.den, osoba1.datumNarodenia.mesiac,
       osoba1.datumNarodenia.rok);
20     return 0;
21 }
```

Úlohy na precvičenie:

Úloha: Definujte štruktúru Bod. Následne túto štruktúru použite pre definíciu tvaru Obdĺžnik (pomocou 4 bodov). Nad touto štruktúrou definujte a aplikujte (zavolajte) nasledujúce funkcie:

- Vzdialenosť dvoch bodov (pomocná funkcia)

```
float vzdialenostBodov(Bod a, Bod b)
```

- obvod obdĺžnika

```
float obvodObdlznika(Obdlznik o)
```

- plocha (obsah) obdĺžnika

```
loat plochaObdlznika(Obdlznik o)
```

- Je zadaný obdĺžnik štvorec?

```
bool jeStvorec(Obdlznik o)
```

- Bod X, leží v obdĺžniku?

```
bool bodLeziVObdlzniku(Obdlznik o, Bod p)
```

- Zmeň veľkosť každej strany obdĺžnika o `int x`

```
Obdlznik zmenVelkostObdlznika(Obdlznik o, int
x)
```