

**Microsoft®**



Beáta Bednárová

# Algoritmizácia v jazyku C

 **OD** študentov  
**PRE** študentov

 Microsoft®  
**Visual Studio® 2008**

Autorka: Beáta Bednárová  
Odborný garant: Ing. Ján Hanák, MVP

### **Algoritmizácia v jazyku C**

Praktické cvičenie zo série „Od študentov pre študentov“

#### **Charakteristika praktických cvičení zo série „Od študentov pre študentov“**

Sme presvedčení o tom, že ak inteligentní mladí ľudia ovládnu najmodernejšie počítačové technológie súčasnosti, ich tvorivý potenciál je vskutku nekonečný. Primárnym cieľom iniciatívy, ktorá stojí za novo uvádzanou sériou praktických cvičení „Od študentov pre študentov“, je maximalizácia hodnôt ľudských kapitálov študentov ako hlavných členov akademických komunít. Praktické cvičenia zo série „Od študentov pre študentov“ umožňujú študentom využiť ich existujúce teoretické znalosti, pričom efektívnym spôsobom predvádzajú, ako možno tieto znalosti s výhodou uplatniť pri vývoji atraktívnych počítačových aplikácií v rôznych programovacích jazykoch (C, C++, C++/CLI, C#, Visual Basic, F#). Budeme nesmierne šťastní, keď sa našim praktickým cvičeniam podarí u študentov prebudiť a naplno rozvinúť záujem o programovanie a vývoj počítačového softvéru. Veríme, že čím viac sofistikovaných IT odborníkov vychováme, tým viac budeme môcť spoločnými silami urobiť všetko pre to, aby sa z tohto sveta stalo lepšie miesto pre život.

# Algoritmizácia v jazyku C

## Praktické cvičenie zo série „Od študentov pre študentov“

Cieľové publikum: **študenti**, ovládajúci základy jazyka C

Vedomostná náročnosť: ☒ ☐ ☐ ☐

Časová náročnosť: **1 hodina a 15 minút**

Programovacie jazyky: **C** (podľa ISO štandardu C90)

Vývojové prostredia: **Visual C++ 2008 Express**, Visual C++ 2010 Express

Operačné systémy: **Windows Vista**, Windows 7, Windows XP

Technológie: **knižnica jazyka C**, exekučné prostredie jazyka C (CRT)



Programovací jazyk C je najpopulárnejším prostriedkom súčasnosti na vývoj štruktúrovaných programov. Hoci od jeho vynájdenia uplynulo už viac ako 30 rokov, magické „céčko“ sa teší neutíchajúcej obľube, a to nielen v akademicknej, ale aj v komerčnej sfére. Takmer každá vysoká škola informatického zamerania má vo svojich študijných programoch výučbové kurzy, ktoré sa venujú algoritmizácii a vývoji počítačového softvéru v jazyku C. Jazyk C má veľa konkurenčných výhod: je to jazyk strednej úrovne, produktom jeho prekladača sú kompilované programy s natívnym (strojovým) kódom, disponuje intuitívnou syntaxou, napomáha rozvíjať abstraktné myslenie študentov a v neposlednom rade pozitívne ovplyvňuje ďalšie významné programovacie jazyky, najmä C++ a C#.

V tomto praktickom cvičení vás zoznámime s tvorbou štruktúrovaných programov rôznej zložitosti v jazyku C. Pri praktickom programovaní budeme využívať vývojové prostredie Microsoft Visual C++ 2008 Express, ktoré je zdarma k dispozícii pre každého záujemcu. Ak Visual C++ 2008 Express nemáte nainštalovaný na svojom počítači, môžete si ho prevziať z nasledujúcich stránok spoločnosti Microsoft: <http://www.microsoft.com/express/download/>.



**Poznámka:** Aby ste mohli z praktického cvičenia vyťažiť maximum, predpokladáme, že ovládáte základy programovania v jazyku C. Nebudeme sa teda zaoberať vysvetľovaním elementárnych princípov a programovacích elementov, ako sú premenné, operátory, rozhodovacie príkazy či cykly. Naším cieľom je využiť vašu existujúcu bázu znalostí pri budovaní prakticky orientovaných programov v jazyku C.

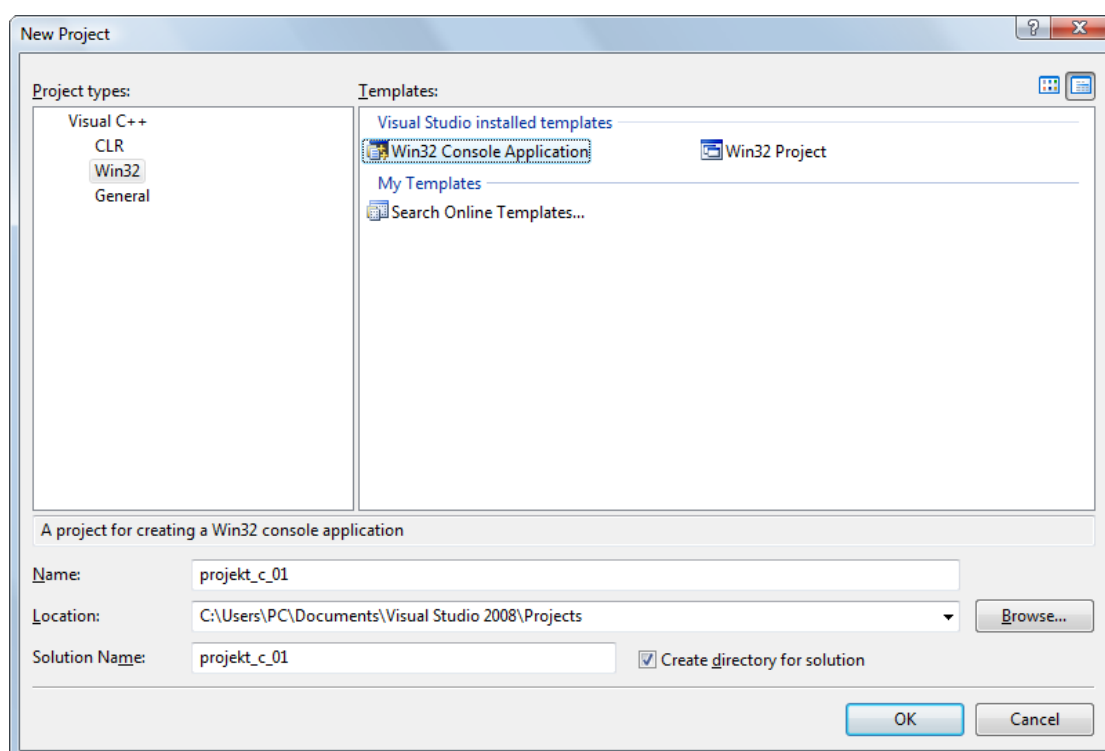
## Obsah praktického cvičenia

1 Založenie nového projektu jazyka C vo vývojovom prostredí Visual C++ 2008 Express .....	3
2 Pridanie zdrojového súboru jazyka C do projektu .....	4
3 Praktický program č. 1: Diagnostika ejekčnej frakcie srdca pacienta .....	7
3.1 Zostavenie programu jazyka C vo vývojovom prostredí Visual C++ 2008 Express .....	9
3.2 Spustenie zostaveného programu jazyka C z vývojového prostredia Visual C++ 2008 Express .....	9
4 Praktický program č. 2: Analytická geometria .....	10
5 Praktický program č. 3: Analýza základných štatistických ukazovateľov dátového súboru .....	13

## 1 Založenie nového projektu jazyka C vo vývojovom prostredí Visual C++ 2008 Express

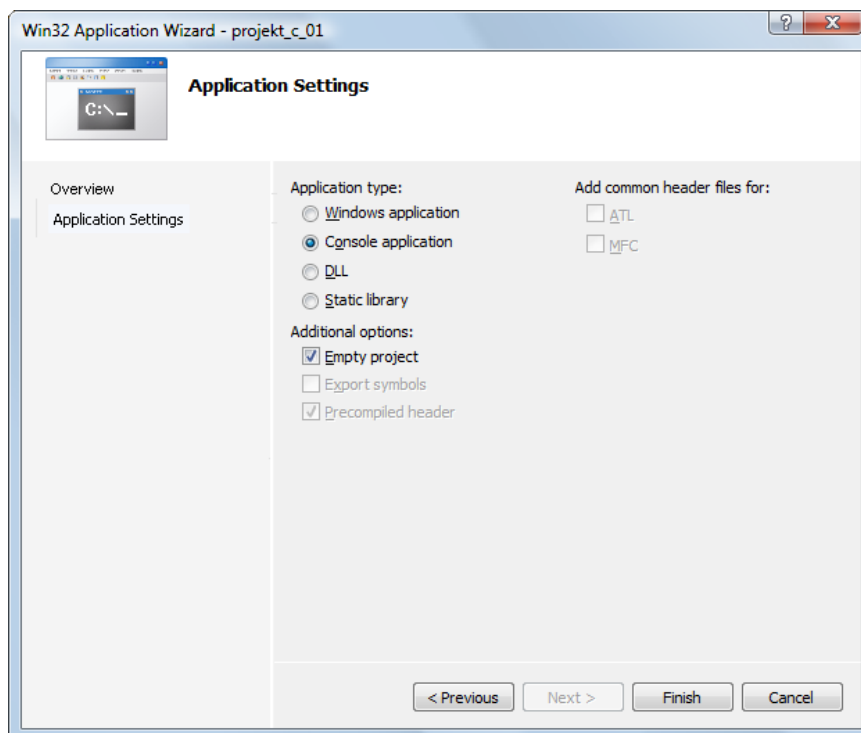
Nový projekt štandardnej konzolovej aplikácie jazyka C založíme vo vývojovom prostredí Visual C++ 2008 Express nasledujúcim spôsobom:

1. Na úvodnej stránke **Start Page** klepneme na hypertextový odkaz **Create Project**.
2. V dialógovom okne **New Project** sa zameriame na stromovú štruktúru **Project types**, z ktorej vyberieme položku **Win32**.
3. V sekcii **Templates** zvolíme projektovú šablónu **Win32 Console Application**.
4. Do textového poľa **Name** zadáme názov projektu. (Visual C++ 2008 Express automaticky vyplní aj textové pole **Solution Name**, a to tak, aby malo riešenie rovnaký názov ako projekt, ktorý bude v riešení uložený.) V tejto chvíli by malo dialógové okno **New Project** vyzeráť ako na obr. 1.



Obr. 1: Založenie nového projektu štandardnej konzolovej aplikácie jazyka C

5. Klepneme na tlačidlo **OK**.
6. Spustí sa sprievodca založením projektu štandardnej konzolovej aplikácie **Win32 Application Wizard**. Klepneme na tlačidlo **Next** a v druhom kroku aktivujeme voľbu **Empty project** pre založenie nového prázdneho projektu (obr. 2).



Obr. 2: Aktivácia voľby **Empty project** pre založenie prázdneho projektu

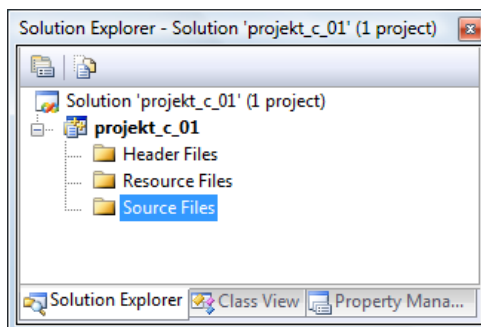
Prázdny projekt je pre nás vyhovujúcim riešením, pretože zdrojový súbor jazyka C pridáme do projektu sami.

- Po klepnutí na tlačidlo **Finish** vytvorí sprievodca nový projekt. Keďže je projekt prázdny, musíme doň pridať jeden zdrojový súbor jazyka C.

## 2 Pridanie zdrojového súboru jazyka C do projektu

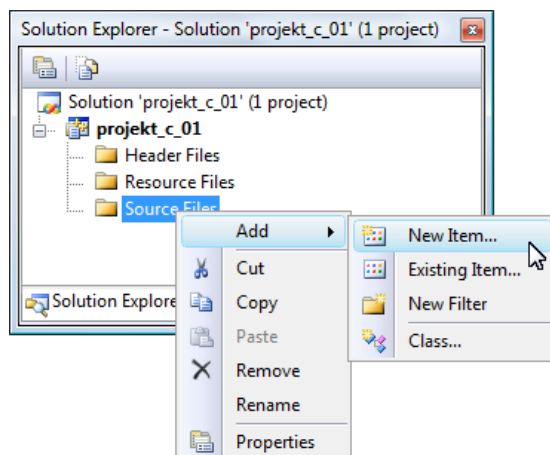
V každom projekte jazyka C sa musí nachádzať aspoň jeden zdrojový súbor jazyka C. V tomto zdrojovom súbore bude uložený zdrojový kód štruktúrovaného programu jazyka C. Zdrojový súbor jazyka C pridáme do nášho projektu takto:

- V podokne **Solution Explorer** (ktoré je štandardne ukotvené pri ľavej strane vývojového prostredia Visual C++ 2008 Express) klepneme pravým tlačidlom myši na priečinok **Source Files** (obr. 3).



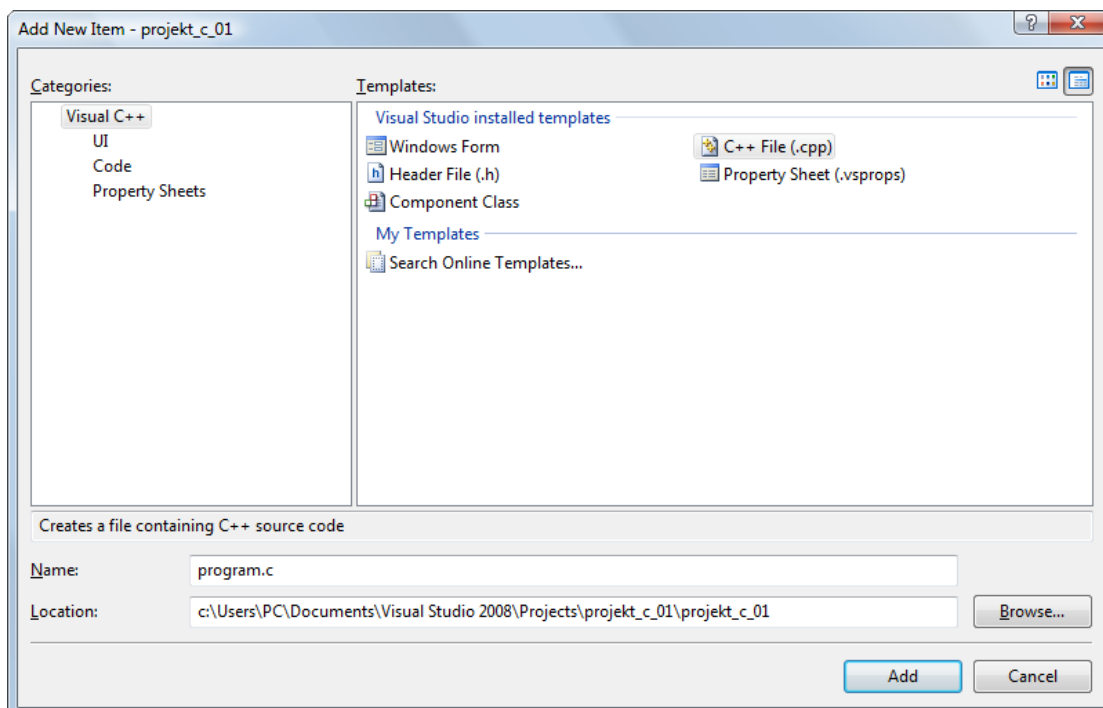
Obr. 3: Pridanie zdrojového súboru jazyka C – 1. fáza

2. Z miestnej ponuky vyberieme príkaz **Add** → **New Item**, čím zviditeľníme dialógové okno pre pridanie novej projektovej súčasti (obr. 4).



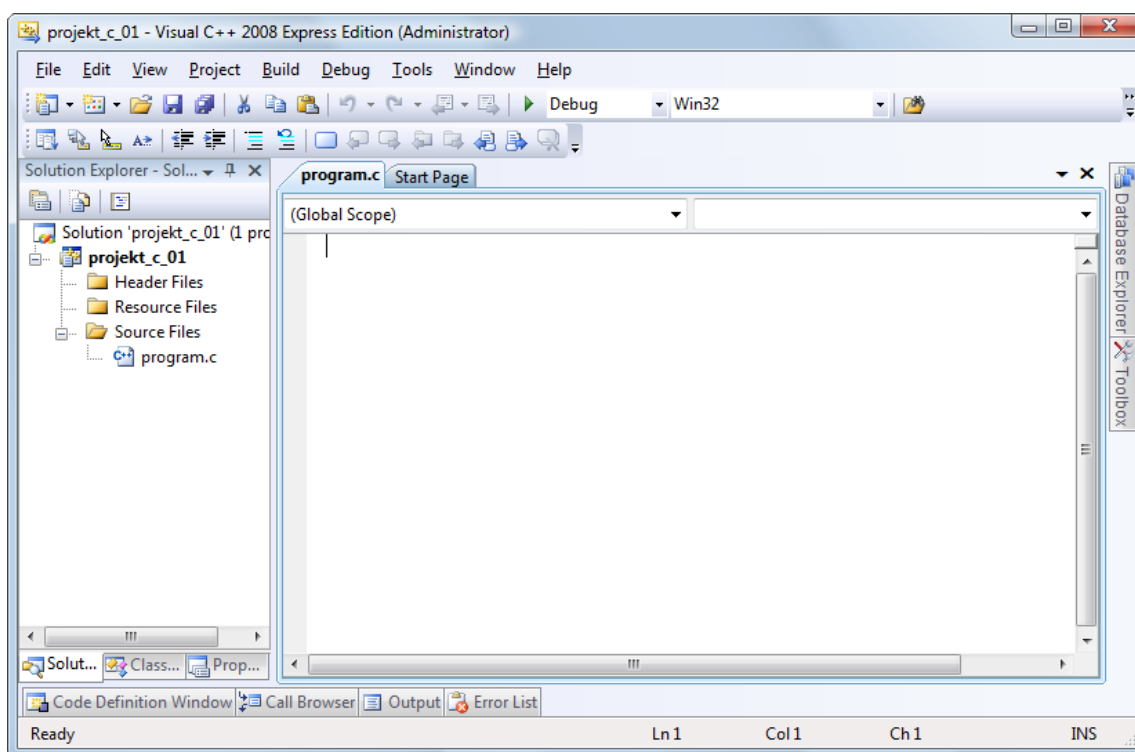
Obr. 4: Pridanie zdrojového súboru jazyka C – 2. fáza

3. V dialógu **Add New Item** zvolíme v sekcii **Templates** súborovú šablónu **C++ File (.cpp)**, ktorá reprezentuje zdrojové súbory jazykov C a C++.
4. Do textového poľa **Name** zapíšeme názov zdrojového súboru jazyka C. Na tomto mieste je dôležité podotknúť, že za názvom zdrojového súboru musí byť explicitne uvedená aj prípona .c, ktorá vraví, že chceme pridať zdrojový súbor jazyka C. Ak by sme príponu .c neuviedli, Visual C++ 2008 Express by implicitne predpokladal, že chceme do projektu pridať zdrojový súbor jazyka C++, a preto by automaticky použil koncovku .cpp (obr. 5).

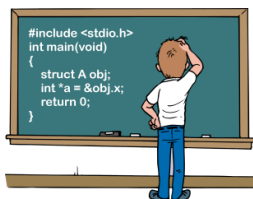


Obr. 5: Pridanie zdrojového súboru jazyka C – 3. fáza

5. Po klepnutí na tlačidlo **Add** sa zdrojový súbor jazyka C pridá do projektu. Obsah novo pridaného zdrojového súboru Visual C++ 2008 Express okamžite otvorí v editore zdrojového kódu (obr. 6).



Obr. 6: Zdrojový súbor je pripravený na zápis zdrojového kódu jazyka C



### 3 Praktický program č. 1: Diagnostika ejekčnej frakcie srdca pacienta

Vedomostná náročnosť:

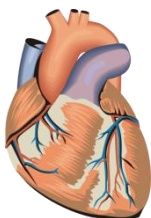
Časová náročnosť: 15 minút

Prvý program bude pôsobiť ako medicínsky softvér, ktorý nám umožní zistiť ejekčnú frakciu srdca pacienta. Ak ste niekedy absolvovali kardiologické vyšetrenie, tak vám lekár pomocou diagnostických nástrojov zisťoval aj ejekčnú frakciu vášho srdca. Ejekčná frakcia je ukazovateľ, ktorý vraví, ako dobre si vaše srdce plní svoju čerpaciu funkciu. Na určenie ejekčnej frakcie srdca použijeme nasledujúci matematický vzorec:

$$E_f = \frac{O_1 - O_2}{O_1} * 100 [\%]$$

kde:

- $E_f$  je ejekčná frakcia v percentuálnom vyjadrení.
- $O_1$  je objem krvi v ľavej srdcovej komore pred kontrakciou srdcového svalu.
- $O_2$  je objem krvi v ľavej srdcovej komore po kontrakcii srdcového svalu.



**Praktický príklad:** V ľavej srdcovej komore sa pred kontrakciou srdcového svalu nachádza 120 ml krvi. Keď sa srdcový sval stiahne, vypudí z ľavej srdcovej komory 70 ml krvi. Po kontrakcii srdcového svalu zostane v ľavej srdcovej komore 50 ml krvi. Aká je ejekčná frakcia pacientovho srdca?

Po dosadení do všeobecného matematického vzorca dostávame:

$$E_f = \frac{O_1 - O_2}{O_1} * 100 = \frac{120 - 50}{120} * 100 = \frac{70}{120} * 100 = 58,33 \%$$

Pacientovo srdce je v poriadku, keď sa percentuálna hodnota ejekčnej frakcie pohybuje v intervale <55 %, 75 %>. V prípade nášho imaginárneho pacienta je ukazovateľ ejekčnej frakcie približne 58 %, čo znamená, že pacientovo srdce dosahuje spodnú hranicu normy pre ejekčnú frakciu.

#### Praktická algoritmizácia programu v jazyku C:

```
#include <stdio.h>

int main(void) {
    /* Definície premenných. */
    int objemKrvPredKontrakciou, objemKrvPoKontrakcii;
    float ejekcnaFrakcia;
    /* Načítanie a uchovanie hodnôt na výpočet ejekčnej frakcie. */
    printf("Zadajte objem krvi v ľavej srdcovej "
           "komore pred kontrakciou srdcového svalu: ");
    scanf("%d", &objemKrvPredKontrakciou);
    printf("Zadajte objem krvi v ľavej srdcovej "
           "komore po kontrakcii srdcového svalu: ");
    scanf("%d", &objemKrvPoKontrakcii);
    /* Výpočet ejekčnej frakcie srdca. */
    ejekcnaFrakcia = (objemKrvPredKontrakciou - objemKrvPoKontrakcii)
        / (float)objemKrvPredKontrakciou * 100;
    /* Zobrazenie diagnostikovanej ejekčnej frakcie srdca na výstupe. */
    printf("Ejekcna frakcia pacientovho srdca je %.2f %%.", ejekcnaFrakcia);
    return 0;
}
```





**Verbálne zhrnutie programu:** Syntakticky sa program skladá z troch častí. V prvej definujeme premenné, do ktorých neskôr uložíme dáta, s ktorými budeme v programe pracovať. Premenné **objemKrviPredKontrakciou** a **objemKrviPoKontrakcii** sú vybavené primitívnym celočíselným dátovým typom **int**, a preto budeme očakávať, že vstupné dáta budú rovnako celočíselnej povahy. Na druhej strane, premennej **ejekcnaFrakcia** sme prisúdili reálny dátový typ **float** s jednoduchou presnosťou. To je v poriadku, pretože ejekčná frakcia je percentuálna hodnota. V druhej časti programu voláme funkcie **printf** a **scanf**, aby sme inštruovali používateľa a načítali od neho vstupné dáta. Napokon, v tretej časti programu vypočítavame hodnotu ejekčnej frakcie a zobrazujeme ju na výstupe.

Kritickým miestom programu je priradovací príkaz, v ktorom dochádza k inicializácii premennej **ejekcnaFrakcia**. Pre lepšiu prehľadnosť uvedieme tento príkaz znova:

```
/* Výpočet ejekčnej frakcie srdca. */
ejekcnaFrakcia = (objemKrviPredKontrakciou - objemKrviPoKontrakcii)
    / (float)objemKrviPredKontrakciou * 100;
```

Prekladač bude tento príkaz spracúvať takto:

1. krok: vyhodnotenie 1. aritmetického podvýrazu.

```
ejekcnaFrakcia = (objemKrviPredKontrakciou - objemKrviPoKontrakcii)
    / (float)objemKrviPredKontrakciou * 100;
```

2. krok: explicitná typová konverzia hodnoty premennej **objemKrviPredKontrakciou** a vyhodnotenie 2. aritmetického podvýrazu.

```
ejekcnaFrakcia = (objemKrviPredKontrakciou - objemKrviPoKontrakcii)
    / (float)objemKrviPredKontrakciou * 100;
```



**Upozornenie:** Explicitnou typovou konverziou meníme dátový typ hodnoty uloženej v premennej **objemKrviPredKontrakciou** z **int** na **float**. To je významné, pretože chceme spracovanie reálneho a nie celočíselného delenia.

3. krok: vyhodnotenie celého aritmetického výrazu.

```
ejekcnaFrakcia = (objemKrviPredKontrakciou - objemKrviPoKontrakcii)
    / (float)objemKrviPredKontrakciou * 100;
```

4. krok: priradenie inicializačnej hodnoty do premennej **ejekcnaFrakcia**.

```
ejekcnaFrakcia = (objemKrviPredKontrakciou - objemKrviPoKontrakcii)
    / (float)objemKrviPredKontrakciou * 100;
```

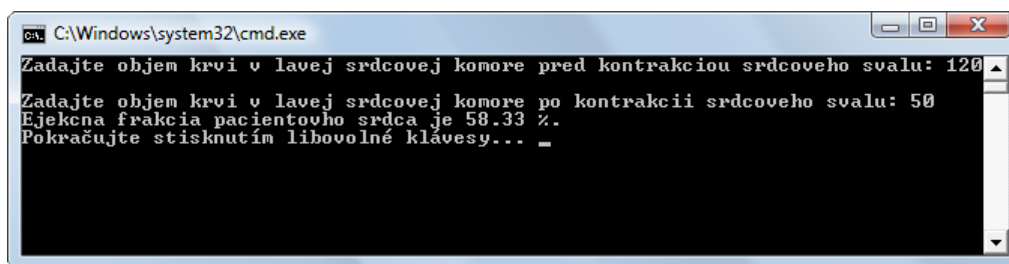
### 3.1 Zostavenie programu jazyka C vo vývojovom prostredí Visual C++ 2008 Express

Vo chvíli, keď máme hotový zdrojový kód programu, môžeme program zostaviť (preložiť, prepojiť a vygenerovať priamo spustiteľný súbor). To urobíme nasledujúcim spôsobom:

1. Otvoríme ponuku **Build** a aktivujeme príkaz **Build Solution** alebo príkaz **Build <NázovProjektu>**. Vzhľadom na to, že pracujeme s jednoprojektovým riešením, je jedno, ktorý príkaz uprednostníme. (S výhodou môžeme využiť aj klávesové skratky **F7** či **CTRL+SHIFT+B**).
2. Proces zostavenia programu môžeme sledovať v podokne **Output**. Ak je všetko v poriadku, v podokne **Output** sa nakoniec zobrazí správa *Build: 1 succeeded*.

### 3.2 Spustenie zostaveného programu jazyka C z vývojového prostredia Visual C++ 2008 Express

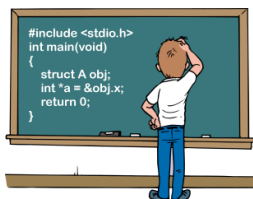
Zostavený program spustíme takto: otvoríme ponuku **Debug** a vyberieme položku **Start Without Debugging** (alebo rýchlejšie použijeme klávesovú skratku **CTRL+F5**). Program sa spustí, pričom máme možnosť preveriť, či produkuje očakávané výstupy (obr. 7).



```

C:\Windows\system32\cmd.exe
Zadajte objem krvi v lavej srdcovej komore pred kontrakciou srdcoveho svalu: 120
Zadajte objem krvi v lavej srdcovej komore po kontrakcii srdcoveho svalu: 50
Ejekčna frakcia pacientovho srdca je 58.33 %.
Pokračujte stisknutím ľubovoľnej klávesy... _
    
```

Obr. 7: Výstup programu, ktorý diagnostikuje ejekčnú frakciu srdca pacienta



## 4 Praktický program č. 2: Analytická geometria

 Vedomostná náročnosť: 

 Časová náročnosť: **20 minút**

Nasledujúci program umožní na základe súradníc dvoch ľubovoľne zadaných bodov  $A[x_1, y_1]$  a  $B[x_2, y_2]$  určiť všeobecnú rovnicu priamky (pretože pre každé dva body existuje práve jedna priamka, ktorá obidvoma prechádza), vzdialenosť zadaných bodov a priesečník priamky s osami v dvojrozmernej súradnicovej sústave. Všeobecná rovnica priamky spolu s parametrickou a smernicovou rovnicou slúži na analytické vyjadrenie priamky.

**Všeobecná rovnica priamky** má tvar:

$$ax + by + c = 0$$

V prvom rade si treba uvedomiť, že na vyjadrenie všeobecnej rovnice priamky potrebujeme poznať:

- Jeden bod ležiaci na priamke.
- Súradnice normálového vektora priamky:  $\vec{n} = [a, b]$ .

Body ležiace na priamke poznáme dva, a to bod  $A[x_1, y_1]$  a  $B[x_2, y_2]$ , ktoré boli zadané. Normálový vektor je ľubovoľný vektor kolmý na smernicový vektor priamky. (Vektor si môžeme predstaviť ako orientovaný úsečku, teda úsečku, na ktorej je vyznačený začiatkový a koncový bod.) Najprv určíme súradnice smernicového vektora priamky podľa nasledujúceho matematického vzťahu:

$$\vec{s} = \overrightarrow{AB} = B - A = [x_2, y_2] - [x_1, y_1] = [x_2 - x_1, y_2 - y_1]$$

Vzhľadom na vzájomný kolmý vzťah medzi normálovým a smernicovým vektorom musí byť ich skalárny súčin rovný nule. Vzťah, na základe ktorého vieme určiť súradnice normálového vektora zo smernicového vektora, je takýto:

$$\vec{n} = [a, b]$$

kde:

- $a = y_2 - y_1$ .
- $b = -(x_2 - x_1)$ .

**Vzdialenosť bodov** alebo veľkosť vektora vyjadríme podľa vzťahu vychádzajúceho z Pytagorovej vety:

$$|\overrightarrow{AB}| = \sqrt{a^2 + b^2}$$

**Priesečníky s osami** v dvojrozmernej súradnicovej sústave majú tieto charakteristiky:

- Priesečník s osou x: bod  $X[x, 0]$ .
- Priesečník s osou y: bod  $Y[0, y]$ .

**Praktický príklad:** Súradnice začiatkového bodu  $A[x_1, y_1]$  a koncového bodu  $B[x_2, y_2]$  smernicového vektora priamky sú  $A[6, 8]$  a  $B[-3, -7]$ . Musíme nájsť odpovede na 3 otázky:

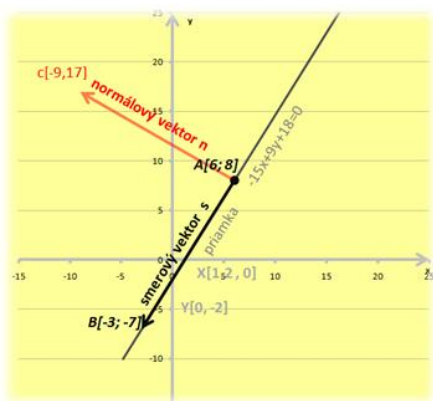
1. Aká je **všeobecná rovnica** priamky?
2. Aká je **vzdialenosť** používateľom definovaných bodov?

### 3. Aké sú **priesečníky** priamky s osami súradnicovej sústavy?

**Všeobecná rovnica priamky:** Na to, aby sme vyjadrili všeobecnú rovnicu priamky, potrebujeme poznať jej normálový vektor, ktorý získame prostredníctvom vyššie uvedeného vzťahu zo smerového vektora:

$$\overrightarrow{AC} = \vec{n} = [a, b] = [y_2 - y_1, x_1 - x_2] = [-7 - 8, 6 + 3] = [-15, 9]$$

Dostaneme neúplnú všeobecnú rovnicu priamky:  $-15x + 9y + c = 0$ .



Pre výpočet koeficienta  $c$  dosadíme do vzťahu súradnice ľubovoľného bodu ležiaceho na priamke. Nami zadané body A a B ležia oba na priamke, preto sa môžeme rozhodnúť pre dosadenie súradníc jedného z nich, napríklad bodu A[6, 8]:

$$\begin{aligned} -15x + 9y + c &= 0 \\ -15 \cdot 6 + 9 \cdot 8 + c &= 0 \\ c &= 18 \end{aligned}$$

**Všeobecná rovnica priamky:  $-15x + 9y + 18 = 0$**

**Vzdialenosť bodov:**

$$|\overline{AB}| = \sqrt{a^2 + b^2} = \sqrt{(-15)^2 + 9^2} = \sqrt{306} = 17,49$$

**Priesečníky:**

- Priesečníkom s osou  $x$  je bod  $X[x, 0]$ .  $X$ -ovú súradnicu bodu  $X$  dopočítame dosadením nulovej  $y$ -ovej súradnice do všeobecnej rovnice priamky:

$$\begin{aligned} -15x + 9y + 18 &= 0 \\ -15x + 9 \cdot 0 + 18 &= 0 \\ x &= 1,2 \end{aligned}$$

- Priesečníkom s osou  $y$  je bod  $Y[0, y]$ . Postupujeme podobne ako pri výpočte predchádzajúceho priesečníka, avšak nulovú súradnicu priradíme  $x$ -ovej súradnici a  $y$ -ovú súradnicu dopočítame takto:

$$\begin{aligned} -15x + 9y + 18 &= 0 \\ -15 \cdot 0 + 9y + 18 &= 0 \\ y &= -2 \end{aligned}$$

Napokon sme vypočítali aj súradnice bodov prieniku priamky  $-15x + 9y + 18 = 0$  s osami súradnicovej sústavy:  $X[1,2; 0]$  a  $Y[0; -2]$ .

## Praktická algoritmizácia programu v jazyku C:

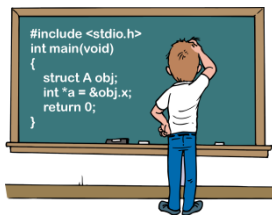
```
#include<stdio.h>
#include<math.h>

int main(void)
{
    /* Definície premenných. */
    float a, b, c;
    float x1, y1, x2, y2;
    float priesečník_x, priesečník_y, vzdialenost;
    /* Načítanie súradníc bodov. */
    printf("Zadajte suradnice prveho bodu: ");
    scanf("%f %f", &x1, &y1);
    printf("Zadajte suradnice druhého bodu: ");
    scanf("%f %f", &x2, &y2);
    /* Výpočet parametra c a súradníc normálového vektora. */
    a = y2 - y1;
    b = -(x2 - x1);
    c = -a * x1 - b * y1;
    /* Výpis všeobecnej rovnice priamky. */
    printf("\nVseobecna rovnica priamky:\n");
    printf("%7.2fx + %7.2fy + %7.2f = 0\n", a, b, c);
    /* Výpočet a výpis vzdialenosti bodov. */
    vzdialenost = (float)sqrt(a * a + b * b);
    printf("\nVzdialenost zadanych bodov: %f\n", vzdialenost);
    /* Dopolčítanie súradníc priesečníkov s osami dvojrozmernej súradnicovej sústavy. */
    priesečník_x = -c / a;
    printf("\nPriesečník s osou x: X[%7.2f,0]\n", priesečník_x);
    priesečník_y = -c / b;
    printf("Priesečník s osou y: Y[0,%7.2f]\n", priesečník_y);
    return 0;
}
```



**Verbálne zhrnutie programu:** Program obsahuje okrem hlavičkového súboru `stdio.h` aj hlavičkový súbor `math.h`, v ktorom sa nachádzajú deklarácie matematických funkcií. Jednou z nich je aj funkcia **`sqrt`**, ktorú v programe používame na výpočet druhej odmocniny. Syntaktický obraz programu možno podobne ako v predchádzajúcom programe rozdeliť do troch celkov. V prvej časti definujeme premenné primitívneho reálneho dátového typu **`float`**. V druhej časti načítavame hodnoty súradníc bodov do premenných **`x1, y1, x2, y2`** pomocou funkcií **`printf`** a **`scanf`**. V tretej časti zabezpečujeme výpočet a výpis všeobecnej rovnice priamky prechádzajúcej bodmi  $A[6, 8]$  a  $B[-3, -7]$ , vzdialenosť bodov a súradnice bodov  $X[x, 0]$  a  $Y[0, y]$ , v ktorých daná priamka pretína osi dvojrozmernej súradnicovej sústavy.

Pri výpise všeobecnej rovnice priamky a priesečníkov je reálnym hodnotám prisúdený formátový špecifikátor **`%7.2f`**, ktorý predstavuje maximálnu dĺžku 7 číselných pozícií, z toho 2 pozície pre desatinné miesta.



## 5 Praktický program č. 3: Analýza základných štatistických ukazovateľov dátového súboru

Vedomostná náročnosť:

Časová náročnosť: **40 minút**

Posledný praktický príklad je štatistickým programom umožňujúcim výpočet základnej miery variability, štandardnej odchýlky, respektíve rozptylu. Nasledujúci program poslúži taktiež na zistenie aritmetického priemeru, mediánu, maximálnej a minimálnej hodnoty čísel z vopred pripraveného súboru dát.

**Aritmetický priemer** je základnou charakteristikou úrovne, ktorú vyjadrujeme ako podiel súčtu hodnôt  $x_i$  znaku X a ich počtu  $n$ . Označujeme ho symbolom  $\bar{x}$ .

Výpočet aritmetického priemeru:

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \dots + x_n}{n} = \frac{\sum_{i=1}^n x_i}{n}$$

**Rozptyl** patrí spolu so štandardnou odchýlkou medzi absolútne miery variability a vyjadruje aritmetický priemer štvorcov (druhých mocnín) odchýlok zadaných hodnôt  $x_i$  znaku X od ich aritmetického priemeru  $\bar{x}$ . Označuje sa symbolom  $s^2$ . Čím je rozptyl väčší, tým sa zadané hodnoty viac odchyľujú od priemeru. V nezoradenom súbore určíme rozptyl nasledujúcim matematickým vzťahom:

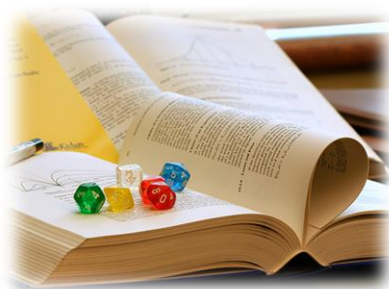
$$s^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

kde:

- $x_i$  sú zadané hodnoty znaku X.
- $n$  je rozsah súboru.
- $\bar{x}$  je aritmetický priemer.

**Štandardnú odchýlku** vyjadríme ako druhú odmocninu z rozptylu hodnôt. Aby sa odstránil vplyv umocňovania, rozptyl sa odmocní, čím sa vypočíta štandardná odchýlka vyjadrujúca rozdiel medzi hodnotami a priemerom pri ignorovaní znamienok:

$$s = \sqrt{s^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2}$$



**Medián** je najpoužívanější kvantil (konkrétne kvantil deliaci súbor na dve časti). Okrem mediánu sa veľmi často používajú kvartily (súbor sa rozdeľuje na štyri časti), decily (súbor sa rozdeľuje na desať častí) a percentily (súbor sa rozdeľuje na sto častí). Medián označujeme symbolom  $\tilde{x}$  a predstavuje hodnotu, ktorá rozdeľuje súbor zadaných hodnôt  $x_1, x_2, \dots, x_n$  usporiadaných vzostupne podľa veľkosti na dve rovnako početné časti. To znamená, že najprv sa musia hodnoty zoradiť od najmenšej po najväčšiu. Mediánom v takto vzostupne usporiadanom rade je prostredná hodnota

(respektíve aritmetický priemer dvoch prostredných hodnôt). Polovica hodnôt znaku X je menšia alebo rovná ako medián, zatiaľ čo polovica hodnôt znaku X je väčšia alebo rovná ako medián.

Ak je počet hodnôt znaku X:

- **nepárny**, potom medián vypočítame jednoducho ako prostrednú hodnotu:  $r = \frac{n+1}{2}$ .
- **párny**, potom určíme medián ako aritmetický priemer dvoch prostredných hodnôt:

$$\tilde{x} = \frac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2}$$

kde:

- $n$  je rozsah súboru.
- $r$  je poradie mediánu.

**Praktický príklad:** Spotreba vody v 6 rôznych domácnostiach počas sledovaného dňa na osobu je: 96, 45, 70, 68, 72 a 120 litrov. Poznáme teda jednotlivé hodnoty  $x_i$  znaku X (X je spotreba vody), z čoho vieme určiť, že rozsah súboru je 6 a súčet celkovej spotreby vody vo všetkých domácnostiach je 471 litrov za deň. Z týchto skutočností potrebujeme zistiť:

- **jednoduchý aritmetický priemer**,
- **rozptyl a štandardnú odchýlku**,
- **medián**,
- **minimálnu a maximálnu hodnotu** objemu spotrebovanej vody.

**Jednoduchý aritmetický priemer** vyjadruje priemerný objem spotrebovanej vody jednou osobou na deň. Výpočet aritmetického priemeru je podmienený súčtom hodnôt znaku  $x_i$  a rozsahom súboru. To znamená, že ho vypočítame ak súčet objemu spotrebovanej vody všetkými domácnosťami (471) vydáme počtom sledovaných domácností (6):

$$\bar{x} = \frac{x_1 + x_2 + x_3 + \dots + x_n}{n} = \frac{\sum_{i=1}^n x_i}{n} = \frac{96 + 45 + 70 + 68 + 72 + 120}{6} = \frac{471}{6} = 78,5$$

Vieme interpretovať, že jedna osoba spotrebuje v priemere 78,5 litrov vody na deň.

**Štandardná odchýlka:** Na výpočet štandardnej odchýlky  $s$  je rozhodujúce určenie rozptylu  $s^2$ . Rozptyl vyčíslime v dvoch krokoch. Najprv vypočítame súčet odchýlok hodnôt  $x_i$  znaku X (spotreba vody) od vypočítaného aritmetického priemeru umocnených na druhú:

$$\sum_{i=1}^n (x_i - \bar{x})^2 = (96 - 78,5)^2 + (45 - 78,5)^2 + (70 - 78,5)^2 + (68 - 78,5)^2 + (72 - 78,5)^2 = 3375,5$$

Na to, aby sme dopočítali rozptyl, vydáme vypočítanú sumu rozsahom súboru:

$$s^2 = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2 = \frac{1}{6} \times 3375,5 = \frac{3375,5}{6} = 562,58$$

Výpočet štandardnej odchýlky je teraz už elementárnou záležitosťou:

$$s = \sqrt{s^2} = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2} = \sqrt{562,583} = 23,72$$

Variabilitu objemu spotrebovanej vody v domácnosti charakterizuje štandardná odchýlka 23,72 litra na osobu za deň.

**Medián** rozdeľuje postupnosť vzostupne (od najmenšej hodnoty po najväčšiu hodnotu) usporiadaných čísel na dve rovnako početné časti. Z toho vyplýva, že na to, aby sme vedeli túto strednú hodnotu vypočítať, musíme najskôr zadané objemy usporiadať vzostupne, čím získame nasledujúcu číselnú postupnosť: 45, 68, 70, 72, 96, 120.

V našom praktickom príklade je počet hodnôt párny a nevieme presne určiť prostrednú z nich, preto medián určíme ako aritmetický priemer dvoch prostredných hodnôt:

$$\tilde{x} = \frac{x_{\frac{n}{2}} + x_{\frac{n}{2}+1}}{2} = \frac{x_3 + x_4}{2} = \frac{70 + 72}{2} = \frac{142}{2} = 71$$

Vieme vyvodiť záver, že 50 % domácností spotrebuje 71 a menej litrov vody denne na osobu, kým 50 % domácností spotrebuje 71 a viac litrov vody denne na osobu.

**Maximálny a minimálny objem** spotrebovanej vody na deň vyčíslime ľahko zo vzostupne usporiadaného radu čísel. V takto usporiadanom rade bude minimom prvá a maximom posledná hodnota, teda **max = 120 l** a **min = 45 l**.

### Praktická algoritmizácia programu v jazyku C:

```
#include<stdio.h>
#include<math.h>
#include<stdlib.h>

int main(void)
{
    /* Definície premenných. */
    int n, i, j;
    float priemer, rozptyl, pomocna_premenna;
    /* Definičné inicializácie premenných. */
    float druha_mocnina_odchylok = 0, suma = 0;
    /* Definícia smerníkovej premennej. */
    float *pole_cisel;
    /* Načítanie rozsahu súboru. */
    printf("Zadajte rozsah suboru: ");
    scanf("%d", &n);
    /* Vytvorenie dynamického poľa. */
    pole_cisel = (float*)malloc(n * sizeof(float));
    /* Testovanie, či máme k dispozícii dostatočné množstvo pamäte. */
    if(pole_cisel == NULL)
    {
        printf("Nedostatok pamätového priestoru.\n");
        return -1;
    }
    /* Načítanie hodnôt do poľa pomocou cyklu for. */
    for(i = 0; i < n; i++)
    {
        printf("Zadajte %d. hodnotu: ", i + 1);
        scanf("%f", &pole_cisel[i]);
    }
    /* Výpočet súčtu zadaných hodnôt a priemeru. */
    for(i = 0; i < n; i++)
    {
        suma += pole_cisel[i];
    }
    priemer = suma / n;
    /* Vyčíslenie štvorcov odchýlok zadaných hodnôt od ich aritmetického priemeru. */
    i = 0;
    while(i < n)
    {
        druha_mocnina_odchylok = druha_mocnina_odchylok +
```



```

        (float)pow((pole_cisel[i] - priemer), 2);
        i++;
    }
    /* Výpočet rozptylu. */
    rozptyl = druha_mocnina_odchylk / n;
    /* Vzostupné usporiadanie čísel pomocou bublinkového zoradenia. */
    for(i = 0; i < n; i++)
    {
        for(j = n - 1; j > i; j--)
        {
            if(pole_cisel[j - 1] > pole_cisel[j])
            {
                pomocna_premenna = pole_cisel[j - 1];
                pole_cisel[j - 1] = pole_cisel[j];
                pole_cisel[j] = pomocna_premenna;
            }
        }
    }
    /* Zobrazenie priemeru, rozptylu, štandardnej odchýlky, mediánu,
    minima a maxima na výstupe. */
    printf("priemer: %f\n", priemer);
    printf("rozptyl: %f\n", rozptyl);
    printf("standardna odchylka: %f\n", (float)sqrt(rozptyl));
    if(n % 2 == 0)
        printf("median: %f\n", (pole_cisel[n / 2 - 1] + pole_cisel[n / 2]) / 2);
    else
        printf("median: %f\n", pole_cisel[n / 2]);
    printf("minimalna hodnota: %f\n", pole_cisel[0]);
    printf("maximalna hodnota: %f\n", pole_cisel[n - 1]);
    /* Výpis vzostupne usporiadaných hodnôt. */
    printf("vzostupne usporiadane hodnoty: ");
    for(i = 0; i < n; i++)
    {
        printf("%5.2f, ", pole_cisel[i]);
    }
    printf("\n");
    /* Uvoľnenie dynamického poľa z pamäte. */
    free((void*)pole_cisel);
    return 0;
}

```



**Verbálne zhrnutie programu:** Okrem hlavičkových súborov použitých v predchádzajúcich programoch sa zoznámime s novým hlavičkovým súborom `stdlib.h`. Tento hlavičkový súbor obsahuje deklarácie funkcií, ktoré nám poslúžia na dynamickú alokáciu (**malloc**, **calloc**, **realloc**) a dealokáciu (**free**) pamäťových blokov.

V nasledujúcom programe sa stretneme s novou dátovou štruktúrou, poľom. Pole je dátová štruktúra obsahujúca  $n$  prvkov rovnakého dátového typu. Jednotlivé prvky poľa sú uložené v pamäti bezprostredne za sebou v ucelenom pamäťovom bloku, pričom sú indexované od 0 po  $n - 1$ . Definícia jednorozmerného automatického poľa vyzerá v jazyku C takto:

```
float automaticke_pole[4];
```

Tento definičný príkaz zakladá nové automatické pole s názvom **automaticke\_pole**. Rozsah poľa je 4, pretože pole obsahuje štyri prvky typu **float**. Jednotlivé prvky poľa sú indexované od 0 po 3, a to nasledujúcim spôsobom:

- 1. prvok poľa: `automaticke_pole[0]`,
- 2. prvok poľa: `automaticke_pole[1]`,
- 3. prvok poľa: `automaticke_pole[2]`,
- 4. prvok poľa: `automaticke_pole[3]`.

V nasledujúcom programe nebudeme využívať takto definované automatické pole, ale dynamicky alokované pole, ktoré na rozdiel od automatického poľa poskytuje efektívnejšie využitie pamäťového priestoru v čase spracovania programu. Syntakticky možno program rozčleniť na 5 častí.

**Začiatok programu** obsahuje jednoduché definície a inicializácie premenných na počiatočnú hodnotu 0, a tiež inicializáciu premennej **n** na hodnotu zadanú používateľom. Automatickým poliám a premenným, ktoré sú správne definované, je pridelená pamäť v čase prekladu zdrojového kódu programu. Definované pole už nemôže zmeniť svoju veľkosť. Problémom je, že často nevieme vopred určiť, aké veľké pole budeme potrebovať. Túto situáciu však dokážeme vyriešiť už spomínanou dynamickou alokáciou poľa. Dynamicky alokované pamäťové bloky sú situované v dynamickej pamäťovej oblasti, ktorú nazývame halda. Na dynamickú alokáciu slúži funkcia **malloc** deklarovaná v stdlib.h. Funkcii **malloc** zadáme alokačnú kapacitu pamäťového bloku (meranú v bajtoch), ktorý žiadame prideliť. V našom prípade:

```
pole_cisel = (float*)malloc(n * sizeof(float));
if(pole_cisel == NULL)
{
    printf("Malo pamati.\n");
    return -1;
}
```

Dynamicky alokovaný pamäťový blok, do ktorého môže byť uložená jedna dynamická premenná typu **float**, má kapacitu 4 bajty. Z toho vyplýva, že celkový počet bajtov nami dynamicky alokovaného bloku pamäte závisí od rozsahu súboru dát, s ktorým budeme pracovať. Funkcia **malloc** vráti smerník na začiatok dynamicky alokovaného pamäťového bloku alebo nulový smerník (NULL) v prípade, ak nie je k dispozícii dostatočné množstvo pamäťového priestoru.

```
pole_cisel = (float*)malloc(n * sizeof(float));
if(pole_cisel == NULL)
{
    printf("Malo pamati.\n");
    return -1;
}
```

V prípade úspešnej dynamickej alokácie pamäťového bloku vracia funkcia **malloc** generický smerník **void\***. Tento generický smerník podrobujeme explicitnej typovej konverzii, v ktorej meníme typ smerníka z **void\*** na **float\***. Inými slovami, v priebehu spracovania explicitnej typovej konverzie pôvodne generický smerník meníme na konkrétny, a teda negenerický smerník (často vravíme, že takéto explicitné pretypovanie zbavuje smerník jeho genericity):

```
pole_cisel=(float*)malloc(n * sizeof(float));
```

Pokiaľ nie je dynamicky alokovaná pamäť potrebná, môžeme ju uvoľniť volaním dealokačnej funkcie **free**:

```
free((void*)pole_cisel)
```

**Druhá časť programu** je zameraná na načítanie hodnôt od používateľa a ich umiestnenie do dynamicky alokovaného poľa. Rovnako realizujeme výpočet súčtu zadaných hodnôt použitím cyklov **for**. Cyklus **for** sa skladá z hlavičky a tela. Hlavička cyklu **for** je zložená z troch častí, ktoré sú oddelené bodkočiarkami. Telo cyklu formujú programové príkazy, ktoré sú obklopené zloženými zátvorkami **{}**.

V našom programe vyzerá hlavička cyklu **for** takto:

```
for(i = 0; i < n; i++)
```

kde:

- **i** je riadiaca premenná cyklu.
- **i = 0** je inicializačný výraz, ktorý inicializuje premennú **i** na dolnú hranicu 0.
- **i < n** je rozhodovací výraz, ktorý udáva, že pokiaľ nebude splnená podmienka **i < n**, cyklus bude ukončený, a teda príkazy v jeho tele sa nevykonajú.
- **i++** inkrementačný výraz, ktorý určuje, akým spôsobom sa bude meniť riadiaca premenná cyklu na konci každej úspešnej iterácie tohto cyklu. Výraz **i++** vraví, že riadiaca premenná **i** sa bude zvyšovať po každej iterácii cyklu o 1.

Prvým cyklom **for** sme načítali nami požadované hodnoty, s ktorými budeme pracovať, do poľa:

```
for(i = 0; i < n; i++)
{
    printf("Zadajte %d. hodnotu: ", i + 1);
    scanf("%f", &pole_cisel[i]);
}
```

Druhý cyklus **for** spočíta všetky zadané hodnoty, aby sme následne mohli vypočítať jednoduchý aritmetický priemer:

```
for(i = 0; i < n; i++)
{
    suma += pole_cisel[i];
}
```

Začiatočnou inicializačnou hodnotou premennej **suma** bola 0. Cyklom **for** k tejto hodnote postupne pripočítavame hodnoty všetkých prvkov poľa, čím získame výsledný súčet objemu spotrebovanej vody.



**Poznámka:** Zložený priradovací príkaz

```
suma += pole_cisel[i];
```

môžeme ekvivalentne nahradiť príkazom s jednoduchým priradením:

```
suma = suma + pole_cisel[i];
```

Obidva z uvedených syntaktických zápisov sú rovnocenné, pretože postupne pripočítavajú všetky hodnoty načítané v predchádzajúcich iteráciách cyklu **for** k hodnote premennej **suma**.

Získaním súčtu hodnôt uzatvárame druhú časť programu tým, že vypočítame aritmetický priemer.

**Tretia časť programu** sa sústreďuje na vyčíslenie druhej mocniny odchýlok zadaných hodnôt  $x_i$  od ich aritmetického priemeru. K výpočtu použijeme cyklus **while**, ktorého všeobecný syntaktický model vyzerá takto:

```
while(podmienka)
{
    príkazy;
}
```

Podmienka je testovaná vždy na začiatku cyklu **while**, pred vykonaním príkazov uložených v jeho tele. V prípade, že je splnená (vyhodnotená na logickú pravdu), uskutočnia sa príkazy v tele cyklu **while**. Ak nebude podmienka splnená (bude vyhodnotená na logickú nepravdu), príkazy v tele cyklu **while** sa nevykonajú a cyklus bude ukončený. Pri algoritmizácii cyklu **while** postupujeme nasledujúcim spôsobom: Najprv inicializujeme premennú **i** na 0, pretože posledná hodnota, ktorá bola do tejto premennej uložená v predchádzajúcom cykle, bola 5. Takto reinitializovaná premenná sa po každej iterácii cyklu **while** navýši o jednotku, až pokiaľ nedosiahne hodnotu menšiu, ako je rozsah súboru.



**Tip:** Cyklus **while** nám postupným načítavaním hodnôt a výpočtom odchýlky načítaného čísla od jeho aritmetického priemeru pomôže pri vyčíslení štandardnej odchýlky. Novinkou použitou v tejto časti programu je matematická funkcia **pow**, ktorá slúži na výpočet  $n$ -tej mocniny čísla. Funkcia **pow** je deklarovaná v hlavičkovom súbore **math.h** nasledujúcim prototypom:

```
double pow(double základ_mocniny, double exponent);
```

Tretia sekcia programu je ukončená výpočtom rozptylu predelením hodnoty premennej **druha\_mocnina\_odchylok** rozsahom súboru **n**.

**Štvrtá časť programu** sa venuje vzostupnému usporiadaniu hodnôt prostredníctvom algoritmu bublinkového zorad'ovania. Bublinkové zorad'ovanie (alebo bubble sort) je jednou z najznámejších metód zorad'ovania programových entít:

```
for(i = 0; i < n; i++)
{
    for(j = n - 1; j > i; j--)
    {
        if(pole_cisel[j - 1] > pole_cisel[j])
        {
            pomocna_premenna = pole_cisel[j - 1];
            pole_cisel[j - 1] = pole_cisel[j];
            pole_cisel[j] = pomocna_premenna;
        }
    }
}
```

Bublinkové zorad'ovanie realizuje vzostupné usporiadanie prvkov. Jeho podstata spočíva v porovnávaní dvoch susediacich hodnôt. Ak nie sú dve susediace hodnoty zoradené vzostupne, to znamená, že hodnota prvku s nižším indexom poľa je väčšia ako hodnota prvku s vyšším indexom poľa, dôjde k ich zámene, ktorú vykonáme prostredníctvom pomocnej premennej. Vezmime si napríklad náš šesťprvkový rad čísel týkajúcich sa dennej spotreby vody: 96, 45, 70, 68, 72, 120. Pri prvej iterácii nadradeného cyklu **for** sa päťkrát spustí vnorený cyklus **for** porovnávajú:

- Prvky s indexmi 4 a 5 (72, 120): ich poradie je správne, preto k zámene nedôjde.
- Prvky s indexmi 3 a 4 (68, 72): poradie je správne, zámena sa neuskutoční.
- Prvky s indexmi 2 a 3 (70, 68): podmienka je splnená, uskutoční sa zámena. Číslo 68 je uložené do prvku s indexom 2, zatiaľ čo do prvku s indexom 3 je priradená číselná hodnota 70.
- Prvky s indexmi 1 a 2 (45, 68): prvky svoje poradie nezmenia.
- Prvky s indexmi 0 a 1 (96, 45). Číslo 45 sa ako najnižšia hodnota presúva na začiatok radu.

Ako vidíme, po 1. iterácii nadradeného cyklu **for** dostávame nasledujúcu postupnosť číselných hodnôt: 45, 96, 68, 70, 72, 120. To znamená, že po spracovaní 1. iterácie nadradeného cyklu **for** prenikne na 1. pozíciu poľa najnižšia hodnota, po vykonaní 2. iterácie tohto cyklu sa na 2. pozíciu dostane v poradí druhé najmenšie číslo atď., až kým nevznikne kompletná postupnosť vzostupne zoradených položiek.

**Posledná časť programu** je zameraná na zobrazenie vypočítaných charakteristík. V prípade, že počet prvkov poľa je párny, medián vypočítame ako aritmetický priemer dvoch prostredných hodnôt. Na konci programu sme použili opäť cyklus **for** na zobrazenie výsledku vzostupného zorad'ovania dát algoritmom bublinkového zorad'ovania. Nakoniec, samozrejme, nesmieme zabudnúť na správnu dealokáciu dynamicky alokovaného poľa.