PROGRAMIRANJE

Predavanje 05 – Polja

Ishod učenja 2

1

OSNOVE TRAŽENJA POGREŠAKA

2

Uvod u *debugging* (ali bez funkcija)

- *Debugging* je postupak kojim tražimo i uklanjamo *bugove* (logičke pogreške) iz programa
- Koraci:
  1. Postaviti jednu ili više točaka prekida (engl. *breakpoint*)
  2. Pokrenuti program sa *Debug* → *Start Debugging* (F5) i pričekati dok izvršavanje stane u točki prekida
  3. Koristiti *Debug* → *Step Over* (F10) i prolaziti liniju po liniju kroz kod, promatrajući vrijednosti varijabli u svakoj liniji
- Ponavljati korak 3 dok ne primijetite da:
  - Vrijednost varijable nije ona koju očekujete (npr, 1 je umjesto 1.5)
  - Sljedeća linija nije ona koju očekujete (npr, ne ulazi u if blok)

ALGEBRA

3

Program 1 – gdje je *bug*?

```
int basic_price = 150000;
int discount = 7500;
int vat = 25;

int dicounted_price = basic_price - discount;
double vat_amount = vat / 100;

double final_price = dicounted_price * (1 + vat_amount);

// Should be: 178.125,00 kn
cout << "Final price is: " << final_price << endl;
```


ALGEBRA

4

Program 2 – gdje je *bug*?

```
int a;
int b;
char operation;

cout << "a: ";
cin >> a;
cout << "b: ";
cin >> b;
cout << "Operation: ";
cin >> operation;

switch (operation) {
   case '+':
      cout << a << " + " << b << " = " << (a + b) << endl;
      break;
   case '-':
      cout << a << " - " << b << " = " << (a - b) << endl;
      break;
   defau1t:
      cout << "Only addition and subtraction are supported" << endl;
      break;
}
```


ALGEBRA

5

POLJA

ALGEBRA

6

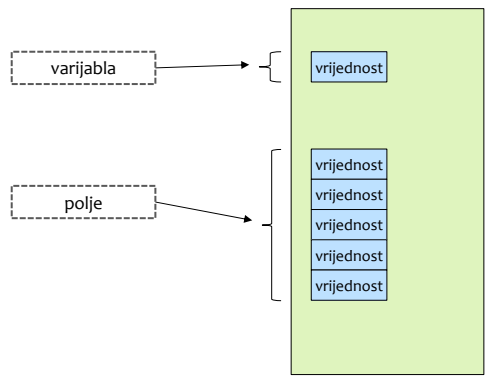
Potreba za poljima

- Recimo da želimo čuvati visinu svih 60 studenata iz učionice
- U ovom trenutku, jedino rješenje je da koristimo 60 varijabli
- Korištenje tako puno varijabli je nespretno i zahtijeva puno tipkanja
  - Primjerice, kako bismo izračunali prosječnu visinu?
- A što ako nemamo 60 već 6.000 vrijednosti?

ALGEBRA

Polje

- Varijabla je dio memorije koji može čuvati jednu vrijednost
- Polje je dio memorije koje može čuvati više vrijednosti
  - 1, 2, 3, 4, 5 ili više vrijednosti



ALGEBRA

Ograničenja polja

- Sve vrijednosti u nekom polju moraju biti istog tipa
- Kad deklariramo polje, moramo koristiti konstantu za određivanje broja elemenata
  - Ne možemo koristiti varijable
  - Veličina polja se koristi u trenutku kompajliranja (engl. *compile-time*), ne u trenutku izvršavanja (engl. *run-time*)
- Polje se smije biti preveliko
  - Varijable i polja su smješteni u posebnom dijelu memorije koji se naziva stog (engl. *stack*)
  - Veličina stoga je samo 1 MB
  - Zbog toga se polja ponekad nazivaju statičkim poljima

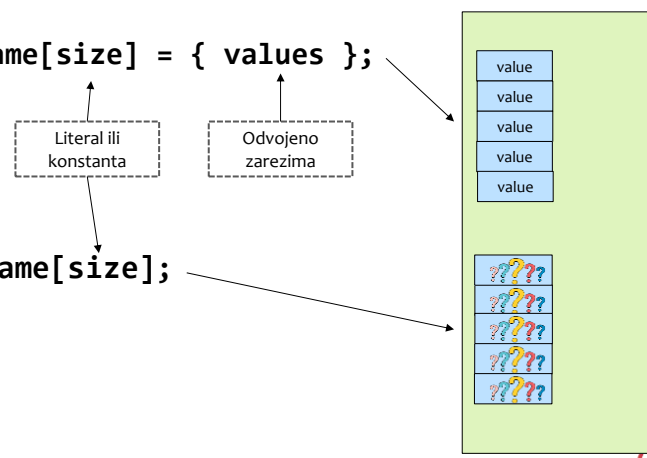
Deklaracija polja

- Deklaracija stvara polje u memoriji
  - Rezervira prostor u memoriji i (opcionalno) upisuje vrijednosti

```
data_type name[size] = { values };
```

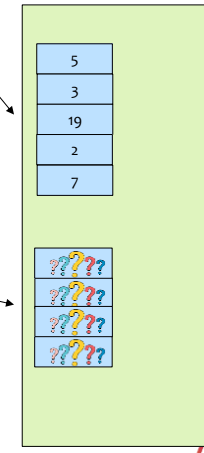
  - ili

```
data_type name[size];
```



Primjeri deklaracija

```
int grades[5] = { 5, 3, 19, 2, 7 };
```




```
int heights[4];
```


ALGEBRA

11

Pristup elementima polja

- Ako deklariramo polje ovako:

```
int numbers[5] = { 10, 20, 30, 40, 50 };
```

- Koristimo sljedeće indekse da pristupimo elementima:

```
numbers[0] // Pristupa elementu 10
numbers[1] // Pristupa elementu 20
numbers[2] // Pristupa elementu 30
numbers[3] // Pristupa elementu 40
numbers[4] // Pristupa elementu 50
```

- Vrijednost u uglatim zagradama se naziva indeks
  - Prvi ispravni indeks je 0, zadnji je size – 1
  - Indeks je udaljenost elementa of početka polja

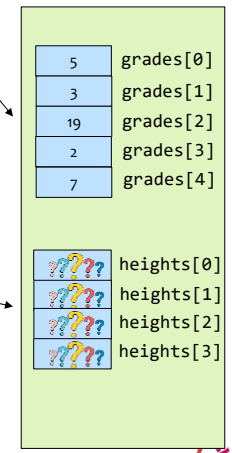
ALGEBRA

12

Pristup elementima polja

```
int grades[5] = { 5, 3, 19, 2, 7 };
```

```
int heights[4];
```



ALGEBRA

Uglate zagrade

- Uglate zagrade smo koristili u dvije situacije:
  1. Kod deklaracije polja smo pisali željenu veličinu polja
  2. Kod pristupa elementu smo pisali željeni indeks elementa
- Nemojmo miješati ove dvije situacije
  - Isti operator (uglate zagrade), ali potpuno drukčije značenje

ALGEBRA

Čitanje i pisanje u elemente polja

- Ključna stvar je uvijek navesti indeks elementa kojem želimo pristupiti
- Pisanje u element polja:

```
numbers[3] = 289;
```

- Čitanje elementa polja:

```
cout << numbers[3];
```


15

Posebna veza polja i petlje for

- Iako polja možemo koristiti bez petlje for, često je jako korisno koristiti ih zajedno
- Koristimo petlju kako bismo kreirali ispravne indekse polja
- Primjer 1

```
int lottery[7] = { 1, 2, 6, 12, 21, 22, 32 };
for (int i = 0; i < 7; i++) {
    cout << lottery[i] << endl;
}
```

- Primjer 2

```
int your_numbers[7];
for (int i = 0; i < 7; i++) {
    cout << "Enter number: ";
    cin >> your_numbers[i];
}
```


16

Veličina polja*

- Operator **`sizeof()`** vraća veličinu u bajtovima
  - `sizeof(`*`variable`*`)` vraća veličinu varijable
  - `sizeof(`*`data_type`*`)` vraća veličinu tipa podataka
  - `sizeof(`*`array`*`)` vraća veličinu polja
  - Ako želimo broj elemenata u polju, moramo podijeliti veličinu polja s veličinom jednog elementa:

```
int numbers[5];
cout << sizeof(numbers); // Prikazuje 20.
cout << sizeof(int); // Prikazuje 4.
cout << sizeof(numbers) / sizeof(int); // Prikazuje 5.
```


Veza između `string` i `char`

- Svaki `string` se može promatrati kao polje `char`-ova, npr:

```
string name = "Ivana";
for (unsigned int i = 0; i < name.length(); i++) {
    char c = name[i];
    cout << c << endl;
}
```

- Metoda `length()` vraća broj `char`-ova u stringu (povratni tip podataka je `unsigned int`)

Primjeri

1. Napravite polje od 5 cjelobrojnih elemenata. Učitajte od korisnika elemente polja i ispišite ih.
2. Napišite program koji izrađuje polje s 5 stringova i odmah im dodjeljuje vrijednosti. Ispišite elemente polja od zadnjeg do prvog.
3. Napišite program koji izrađuje polje sa 7 decimalnih elemenata i odmah im dodjeljuje vrijednosti. Ispišite elemente polja u istom redu. Iza svakog broja stavite " | ", ali iza zadnjeg broja nemojte staviti ništa.

ALGEBRA

19

Primjeri

4. Napišite program koji u polje upisuje brojeve od 1 do 1000. Neka program ispiše sve elemente polja koji su djeljivi sa 7 i 11.
5. Napišite program koji od korisnika učitava 10 cjelobrojnih elemenata u polje. Neka program nakon toga pronađe najmanji element, najveći element i aritmetičku sredinu elemenata u polju i ispiše te podatke.
6. Napišite program koji definira polje realnih brojeva od 5 elemenata. Neka program učita od korisnika dva realna broja *x* i *y*. Neka program ispiše sve brojeve iz polja čija je vrijednost manja od *x*, a veća od *y* (smatramo da je uvijek *x* > *y*). Npr. za polje brojeva 16.7 2.5 9.5 3.4 7.9 i brojeve *x* = 12.05 i *y* = 7.05, potrebno je ispisati brojeve 9.5 7.9

ALGEBRA

20

Primjeri

7. Zadatak vam je implementirati jednostavno šifriranje teksta. Definirajte polje od 36 char-ova i u njega upišite sva mala slova engleske abecede (26 komada) i brojeve 0 – 9 (10 komada). Definirajte još jedno polje i u njega upišite ista slova i brojeve, ali opadajuće (u prvom polju imamo 'a',…,'z','0',…,'9', a u drugom '9',…,'0', 'z',…,'a').

   Od korisnika učitajte rečenicu i sva njena slova i brojke šifrirajte na način:

   a. Za svaki znak pronađite na kojem se indeksu nalazi u prvom polju. Ako ga nema, ispišite originalni znak.

   b. Uzmite znak na tom istom indeksu, ali u drugom polju i ispišite ga.

ALGEBRA

21

Primjeri

8. Za rečenicu učitanu od korisnika ispišite koliko samoglasnika sadržava. Uzmite u obzir i velika i mala slova.

9. Pripremite polje od 5 cijelih brojeva. Omogućite korisniku unos onoliko brojeva koliko želi, na način da nakon svakog unosa pitate korisnika želi li još, ali samo ako već nije unio 5 brojeva. Nakon završenog unosa ispišite aritmetičku sredinu unesenih brojeva.

ALGEBRA

22

Zadaci za sljedećih 7 dana

- Prije sljedećeg predavanja trebate:
  1. Pročitati iz *Demistificirani C++*:
     - 5.3.1 Klasa vector
  2. Pogledati sljedeće:
     - W06-1 Vectors
       - https://youtu.be/W6_BWjgiqeM

ALGEBRA