Увод в програмирането

Скаларни типове

символен и изброим

Символен тип

- **Символният** типът се нарича още **знаков** и се причислява към интегралните типове на езика. За означаването му се използва запазената дума **char**.
- **Множество от стойности**

  Състои се от крайно и наредено множество от символи, обикновено вариант на ISO-646, например ASCII, и обхваща символите от клавиатурата. Стандартизирана е само част от символното множество – 128 символа (главните и малки буква на латинската азбука, цифрите, символите за пунктуация и някои управляващи символи)

Символен тип …

- Два вида символи – **графични** и **управляващи**.
- Графичните символи имат видимо представяне. Означават се чрез ограждане на символа в апострофи.

  Пример:

  ```
  'a' 'd' 'K' '!' '1' ' ' '+'
  ```

- Графичните символи `\` и `'` се представят чрез `'\\'` и `'\''` съответно. Символът `?` се представя и като `'?'` ,и като `'\?'`.

Символен тип …

- **Управляващите символи** нямат видимо представяне. Означават се по следния начин:

  ‘\символ’

Символ	**Предназначение**
\a	издава звуков сигнал
\b	връща курсора един символ назад
\n	предизвиква преминаване на нов ред
\r	връща курсора в началото на реда
\t	хоризонтална табулация
\v	вертикална табулация
\0	нулев символ, край на низ

Символен тип …

- Възможно е и следното означение:
  ‘\nnn’
  където nnn е кодът (ASCII) на символ

Пример:

```
'\32'    ‘\0x20’  ‘ ‘      интервал
‘\10’    ‘\0x0A’  ‘\n’     нов ред
‘\65’    ‘\0x41’  ‘A’
‘\97’    ‘\0x61’  ‘a’
```

Символен тип …

- Елементите от множеството от стойности на типа символен са константите на този тип. Наричат се още **символни литерали**
- Променлива величина, множеството от допустимите стойности, на която съвпада с множеството от стойности на типа символен, се нарича **символна променлива** или **променлива от тип символен**. Дефинира се по общоприетия начин.

Символен тип …

- *Примери:*

```
char c1;
char c2 = '+';
```

- Елементите от множеството от стойности на типа char е наредено. Всеки символ е свързан с код, съгласно кодирането ASCII. При това кодиране, за управляващите символи се използват целите числа от 0 до 31 включително. Останалите символи се кодират с числата от 32 до 255

Символен тип …
Операции над символни данни

- ***Намиране на кода на символ***

  Извършва се чрез израза (int)c1, където c1 е символна константа или променлива.
- *Пример:*

```
cout << (int)'F';
```

Символен тип …
Операции над символни данни

- ***Намиране на символ по даден код***

  Осъществява се чрез израза
  (char)<цял_аритметичен_израз>

  Стойността на <цял_аритметичен_израз> трябва да е неотрицателно цяло число. Ако не е от интервала [0, 255], намира се остатъкът от делението по модул 256.

- *Пример:*

```
cout << (char)65 << '\t' << (char)(3*256+65);
```

  извежда два пъти символа A главно, латинско, разделени с табулация.

Символен тип …
Операции над символни данни

- ***Аритметични операции***

  Всички аритметични операции, допустими над целочислени данни са допустими и за данни от тип от тип char. Извършват се над кодовете им. Резултатът е цяло число. Така се разширява синтаксисът на целите изрази.
- *Примери:*

1. c1 + 15 - ‘A’ е цял аритметичен израз.
2. c1%2 е цял аритметичен израз.

Символен тип …
Операции над символни данни

- ***Присвояване на стойност***

  На променлива от символен тип може да се присвояват символни константи и променливи, а също стойностите на цели аритметични изрази. В последния случай, добрият стил на програмиране изисква да се конвертира явно типът на целия аритметичен израз до символен.
- ***Примери:***

```
c1 = 'A';
c2 = c1;
c1 = c1 + c2 - 15;
```

Символен тип …
Операции над символни данни

- ***Логически операции***

  Всички логически операции са допустими и над данни от символен тип. Изпълняват се над кодовете им. Резултатът е от булев тип. Освен това, символна константа или променлива, поставена на място на условие, изпълнява ролята на булев израз.
- ***Операции за сравнение***

  Над данни от тип символен могат да се прилагат стандартните инфиксни оператори за сравнение. Сравняват се кодовете. Резултатът е булев.
- *Примери:*

```
'a' <= 'z'    е true          'a' <= '0'    е false
'0' < '9'     е true          'A' > 'a'     е false
'x' != 'X'    е true

'a' == 0      c1 != 65        c2 >= 122
```

Символен тип …
Операции над символни данни

- ***Въвеждане***
- Осъществява се чрез оператора >>. В този случай, операторът >> не е чувствителен към символите: интервал, хоризонтална и вертикална табулации и преминаване на нов ред.
- *Примери:*

```
cin >> c1 >> c2;

cout << “c1= “;
char c1;
cin >> c1;
cout << “c2= “;
char c2;
cin >> c2;
```

Символен тип …
Операции над символни данни

- ***Извеждане***

  Осъществява се по стандартния начин.
- *Примери:*

```
cout << c1;
cout << c1+c2;
cout << char(c1+c2);
```

Символен тип … Допълнения

1. Типът char е допустим за тип на switch-израз, т.е. допустим е фрагментът:

```
char c;
cin >> c;
switch (c)
{case 'a': …
 case 'e': …
 …
}
```

2. Чрез модификаторите signed и unsigned се получават разновидности на типа char

Тип изброим

- Типът е стандартен. За разлика от другите, досега разгледани типове, той се дефинира от програмиста като се изброяват константите му. Затова се нарича още потребителски дефиниран тип.

Тип изброим …
Дефиниция на тип изброим

- **Дефиниране на тип изброен**

<дефиниция_на_тип_изброен> ::=
enum [<име_на_тип>]
{ <идентификатор$_1$> [= <константен_израз$_1$>],
<идентификатор$_2$> [= <константен_израз$_2$>],
…
<идентификатор$_n$> [= <константен_израз$_n$>]
};

<име_на_тип> ::= <идентификатор>

където

- enum е запазена дума (съкращение от enumerate – изброявам);
- <константен_израз$_i$> (i = 1, 2,…, n) е константен израз от интегрален тип.

Тип изброим …

- *Примери:*

```
enum Weekday{SUNDAY, MONDAY,
 TUESDAY, WEDNESDAY,
 THURSDAY, FRIDAY, SATURDAY};
enum Name{IVAN=5, PETER=3, MERY=8,
  SONIA=6, VERA=10};
enum Id{A1, A2, A3, A4=8, A5, A6=10,
  A7, A8};
enum {FALSE, TRUE};
```

Тип изброим …

- **Множество от стойности**
  Състои се от всички изброени в дефиницията на типа идентификатори.
- *Пример:*

множество от стойности на тип Weekday

- Елементите от множеството от стойности на даден изброим тип са константите на този тип. Затова ще ги означаваме с главни букви.

Тип изброим …

- Променлива величина, приемаща за стойности константи от множеството от стойности на някакъв изброен тип, се нарича **променлива от този изброим тип**. Дефинира се по общоприетия начин.
- *Примери:*

```
Weekday d1, d2 = SUNDAY;
Name a;
Name b = VERA;
Id x = A2, y = A7;
```

Тип изброим …

- **Вътрешните представяния** на константите от изброими типове се определят по следния начин:
  - Ако не са указани стойности, по подразбиране първият идентификатор получава стойност 0, а всеки следващ – стойност с единица по-голяма от стойността на предходния.

    Пример: Weekday

    SUNDAY - 0, MONDAY - 1, TUESDAY – 2 и т.н. SATURDAY – 6.

Тип изброим …

- Ако всички идентификатори са свързани със стойности, вътрешното представяне на всяка константа от такъв изброен тип е указаната стойност.

  Пример: Name

  IVAN - 5, PETER - 3, MARY - 8, SONIA - 6, VERA - 10.
- •Ако някои идентификатори са свързани със стойности, а други – не, вътрешното представяне на идентификатор, за който е указана стойност е указаната стойност, а на всеки идентификатор, за който не е указана стойност – е вътрешното представяне на идентификатора пред него, увеличено с 1. Вътрешното представяне на първият идентификатор, ако не е указана стойност за него, е 0.

Тип изброим … Операции

- ***Намиране на кода на константа от тип изброен***

  Извършва се чрез израза (int)x, където x е константа или променлива от изброен тип.

- *Пример:*

```
cout << (int)FRIDAY;
```

извежда 5 - кода на FRIDAY.

Тип изброим … Операции

- ***Намиране на константа от тип изброен по даден код***

  Осъществява се чрез израза (<име_на_тип_изброен>)<код>.
- *Пример:*

```
d1 = (Weekday)4;
```

  намира THURSDAY.

Тип изброим … Операции

- ***Аритметични операции***

  Всички аритметични операции, допустими над целочислени данни са допустими и за данни от изброим тип. Извършват се над кодовете на данните. Резултатът е цяло число.
- *Примери:*

  d1+d2–4

  е цял аритметичен израз, стойността на който се получава като се съберат кодовете на d1 и d2 и от полученото се извади 4.

  d1 % 5

  е цял аритметичен израз, изразяващ остатъка от делението на кода на d1 на 5.

Тип изброим … Операции

- ***Присвояване на стойност***

  На променлива от изброим тип може да се присвои която и да е константа от множеството от стойности на типа, от който е променливата, а също и стойността на променлива от същия изброим тип.

- *Пример:*

```
d1 = WEDNESDAY;
d2 = d1;
d1 = d2 - 2;     // не е допустимо
d1 = (Weekday)(d2 - 2);  //вече е допустимо.
```

Тип изброим …
Операции

- ***Логически операции***

  Всички логически операции, са допустими и за данни от тип изброен. Извършват се на кодовете на данните. Резултатът е булев.

- ***Операции за сравнение***

  Данни от един и същ изброим тип могат да се сравняват чрез стандартните инфиксни оператори за сравнение. Сравняват се кодовете. Резултатът е булев.

Тип изброим … Операции

- ***Въвеждане***

  Не е възможно въвеждане на стойност на променлива от някакъв изброим тип чрез оператора >>.

- ***Извеждане***

  Операторите

  cout << <константа_от_тип_изброим>;

  или

  cout << <променлива_от_тип_изброим>;

  извеждат кода на константата или променливата.

Тип изброим … Операции

```
...
enum Weekday {SUNDAY, MONDAY, TUESDAY, WEDNESDAY,
       THURSDAY, FRIDAY, SATURDAY};
Weekday d;
d = FRIDAY;
switch(d)
{ case SUNDAY: cout << "SUNDAY \n"; break;
  case MONDAY: cout << "MONDAY \n"; break;
  case TUESDAY: cout << "TUESDAY \n"; break;
  case WEDNESDAY: cout << "WEDNESDAY \n"; break;
  case THURSDAY: cout << "THURSDAY \n"; break;
  case FRIDAY : cout << "FRIDAY \n"; break;
  case SATURDAY: cout << "SATURDAY \n";
}
```