Низове, Тип изброен, Функции

гл.ас. д-р. Нора Ангелова

(като масиви от символи)

- Едномерен масив от символи
- Името на променлива от тип низ е **константен** указател

```
char str[7] = "123456";
while(*str) {
  cout << *str << " ";
  str++; // ERROR
}</pre>
```

(като масиви от символи)

- Едномерен масив от символи
- Името на променлива от тип низ е константен указател

(като масиви от символи)

- Едномерен масив от символи
- Указател към символ от низа

```
char str[7] = "abcdef";
char *p = str;
```

```
cout << p+3; // def
```

Низове (като указатели от тип char)

//123456

(като указател от тип char)

char *str = "123456";

while(*str) {

 cout << *str << " ";

 str++;

указател

```
(като указател от тип char)
char *str = "123456";
не може да бъде заменено от
char *str;
cin >> str; // ERROR
(getline, ...)
!Не е възможно въвеждане на стойност от тип
```

(като указател от тип char)

```
char str[7] = "123456";
*(str+1) = '9';
str[1] = '9';
cout << str; // 193456
char *str2 = "123456";
*(str2+1) = '9'; // ERROR
str2[1] = '9'; // ERROR
cout << str2;</pre>
```

Кой е по-по-най

}

🔾 Дадена е квадратна матрица от низове. Да се напише програма, която намира броя на палиндромите под главния диагонал(заедно с него). //... int br = 0; char a[MAX_SIZE][MAX_SIZE][MAX_SIZE2]; //... for(int i=0; i<MAX_SIZE; i++)</pre> for(int j=0; j<=i; j++) {</pre> int len = strlen(a[i][j]); char rev[MAX_SIZE2]; for(int k=len-1; k>=0; k--) rev[len-k-1] = a[i][j][k] $rev[len] = '\0';$ if (!strcmp(a[i][j], rev)) br++;

Обработка на знакови данни

• Да се напише програмен фрагмент, който намира стойността на правилно записан числов израз без скоби и без приоритет на операторите +, -, * и /. Пресмятането да завършва след въвеждане на знака =

Например 3+4*5

Обработка на знакови

Данни

```
char op = '+'; double result = 0.0;
do
  double arg; cin >> arg;
  switch (op)
    case '+' : result = result + arg; break;
    case '-' : result = result - arg; break;
    case '*' : result = result * arg; break;
    case '/' : if (arg != 0) result = result / arg;
               else {
                  cout << "Error!\n ";</pre>
                  return 1;
  cin >> op;
} while (op != '=');
cout << result << endl;</pre>
```

- Стандартен тип
- Потребителски дефиниран
- При дефиниция се изброяват константите му
- Интегрален тип може да се използва за тип на израз на оператора switch

```
enum <ume на тип> {
  <идентификатор1>=[<константен_израз1>]опц,
  <идентификатор2>=[<константен израз2>]опц,
  • • • •
  <идентификаторn>=[<константен изразn>]опц
Пример:
enum Weekday {SUNDAY, MONDAY, TUESDAY,
     WEDNESDAY, THURSDAY, FRIDAY, SATURDAY);
```

 Множество от стойности – всички изброени в дефиницията идентификатори

Пример:

enum Weekday {SUNDAY, MONDAY, TUESDAY,
WEDNESDAY, THURSDAY, FRIDAY, SATURDAY};

Множество от стойности на Weekday: SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY

- Елементите на множеството от стойности са константи на този тип изброен
- Означават се с главни букви
- Елементите трябва да бъдат различни идентификатори не само в рамките на дефиницията на типа, но и в дефинициите на всички изброени типове в програмата

• Създаване на променлива от тип изброен

```
тип <име_на_променлива> = <константа_от_типа>
```

Пример:

```
enum Weekday {SUNDAY, MONDAY, TUESDAY,
WEDNESDAY, THURSDAY, FRIDAY, SATURDAY};
Weekday a = MONDAY;
```

• Вътрешно представяне

 Всеки идентификатор(константа на изброен тип) е свързан с код

 Вътрешно представяне
 Ако не са указани константни изрази - кодът се задава по подразбиране

```
Пример
enum Weekday {SUNDAY, MONDAY, TUESDAY,
WEDNESDAY, THURSDAY, FRIDAY, SATURDAY};
SUNDAY - 0;
MONDAY - 1;
...
SATURDAY - 6;
```

• Вътрешно представяне

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

```
Пример
```

```
enum Weekday {SUNDAY=7, MONDAY=1, TUESDAY=2,
WEDNESDAY=3, THURSDAY=4, FRIDAY=5,
SATURDAY=6};
```

```
MONDAY - 1;
```

• Вътрешно представяне

Ако някои идентификатори са свързани със стойности, а други – не са

Пример

enum Weekday {SUNDAY=7, MONDAY=1, TUESDAY,
WEDNESDAY, THURSDAY, FRIDAY, SATURDAY=6};

FRIDAY - 5;

Стойностите се определят от стойността на кода на предишния идентификатор +1

• Вътрешно представяне

```
Пример
enum Weekday {SUNDAY=7, MONDAY=1, TUESDAY, WEDNESDAY, THURSDAY=2, FRIDAY, SATURDAY=6};
FRIDAY - 3;
```

• Намиране на код на константа от тип изброен

```
Пример
```

```
enum Weekday {SUNDAY=7, MONDAY=1, TUESDAY,
WEDNESDAY, THURSDAY, FRIDAY, SATURDAY=6};
```

```
cout << (int)FRIDAY; //5</pre>
```

 Намиране на константа от тип изброен по даден код

```
Пример
```

```
enum Weekday {SUNDAY=7, MONDAY=1, TUESDAY,
WEDNESDAY, THURSDAY, FRIDAY, SATURDAY=6};
```

```
(Weekday)5 -> намира FRIDAY cout << (Weekday)5; // 5
```

 Не е възможно въвеждане на стойност на променлива от тип изброен чрез оператора >>

```
Weekday a;
cin >> a; //ERROR
• Извеждане
```

```
cout << <променлива_от_тип_изброен> cout << <константа_от_тип_изброен>
```

Извежда кода на константата или променливата

- Аритметични операции: +,-,*,/,% Резултат - цяло число
- Логически опреации: &&, | |,!
 Резултат булева стойност

Извършват се над кодовете

```
Пример:
Weekday a = FRIDAY;
Weekday b = a;
a = b-2; // !ERROR
a = (Weekday)(b-2); // !CORRECT
```

Оператори за сравнение: <, <=, >, >=, ==, !=

```
Пример:
MONDAY < FRIDAY

Weekday a = FRIDAY;

Weekday b = a;

if (b==a) cout << "equal";
```

Да се напише програмен фрагмент, който извежда с думи константите на типа Weekday

```
Weekday d = FRIDAY;
switch (d)
  case SUNDAY: cout << "SUNDAY \n"; break;</pre>
  case MONDAY: cout << "MONDAY \n"; break;</pre>
  case TUESDAY: cout << "TUESDAY \n"; break;</pre>
  case WEDNESDAY: cout << "WEDNESDAY \n"; break;</pre>
  case THURSDAY: cout << "THURSDAY \n"; break;</pre>
  case FRIDAY : cout << "FRIDAY \n"; break;</pre>
  case SATURDAY: cout << "SATURDAY \n";</pre>
```

Да се напише програмен фрагмент, който намира средната седмична температура. Температурата се въвежда от потребителя.

```
enum Weekday { SUNDAY=7, MONDAY=1, TUESDAY,
WEDNESDAY, THURSDAY, FRIDAY, SATURDAY);
Weekday d;
double s = 0.0;
double t;
for (d = MONDAY; d <= SUNDAY; d = (Weekday)(d+1))</pre>
  cout << "t= "; cin >> t;
  s = s + t;
cout << setprecision(2) << fixed << setw(10) << s/7</pre>
<< "\n";
```

- Основни структурни единици
- Състоят се от множество от оператори, оформени подходящо, за да се използват като обобщено действие или операция
- Функцията се дефинира еднократно и може да бъде изпълнявана многократно за различни входни данни

Програмите се състоят от:

- Една или повече функции.
- Задължително е да има точно една с име main главна функция

Главната функция е първата функция, която се изпълнява при стартиране на програмата.

Предимства:

- Ясни и лесни за тестване програми
- Липса на повторение на код
- Памет

- Не могат да се дефинират в други функции
- Преди да се извика една функция, тя трябва да е декларирана/дефинирана.

Ако функцията main извиква функцията sum, функцията sum трябва да бъде декларирана/дефинирана преди main.

sum

. . .

main

```
Дефиниция:
```

Заглавие<тип_на_функция><име_на_функция>(<формални_параметри>)

тип на функция – име на тип.

Типът на резултата от изпълнението на функцията.

Ако функцията не връща резултат – тип void.

име на функция – идентификатор

формални параметри – множество от параметри. Те изпълняват ролята на входните данни на функцията.

Функцията може да няма формални параметри.

параметър – <име_на_тип><име_на_параметър>

• Тяло – редица от оператори и дефиниции оградени в { }
{ <тяло> }

• Обръщение към функция

```
<ume_на_функция>() |
<ume_на_функция>(<фактически_параметри>)
```

Фактическите параметри трябва да съответстват по брой, тип, вид и смисъл на формалните параметри.

Оператор return return <израз>;

Пресмята се стойността на израза и се конвертира до типа на функцията (ако това е възможно). Получената стойност се връща в мястото на извикването на функцията и се прекратява изпълнението ѝ.

Функция, която връща сбора на две цели числа Пример:

```
int sum(int a, int b) {
  return a + b;
void main()
  int s = sum(2, 3);
  return;
```

Свързване на формални с фактически параметри:

- Формален параметър стойност
- В стековата рамка на функцията за формалния параметър се отделя толкова памет, колкото типът му изисква и в нея се копира стойността на фактическия параметър
- Формален параметър указател

В стековата рамка на функцията за формалния параметър се отделят 4В. В тях се записва стойността на фактическия параметър, който трябва да бъде адрес на променлива.

Указатели

```
void test(int p) {
int a = 5;
test(a)
addr1
```

p

5

Указатели

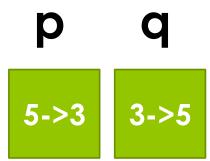
```
void test(int* p) {
int a = 5;
test(&a)
addr1
```

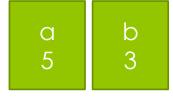
p

addr1

Задача

```
void swap(int p, int q) {
  int temp = p;
  p = q;
 q = temp;
int main()
  int a = 5, b = 3;
  swap(a, b); // 5 3
  return 0;
```





addr1 addr2

Задача

```
void swap2(int* p, int* q) {
  int temp = *p;
  *p = *q;
                                   addr1
                                          addr2
  *q = temp;
int main()
                                    5->3
  int a = 5, b = 3;
                                   addr1 addr2
  swap2(&a, &b); // 3 5
  return 0;
```

cout << "Край";