Упражнение

гл.ас. д-р. Нора Ангелова

Рекурсия

```
// Рекурсивна процедура, намираща всички пътища от
// клетка (x,y) до клетка (7,7) crrway е текущо изминатият път, а l е дължината му.
void way(int x, int y, int *crrWay, int l) {
    crrWay[2*l] = x;
    crrWay[2*l+1] = y;

    // Клетката е извън лабиринта.
    if (x < 0 || y < 0 || x > 7 || y > 7) {
        return;
    }

    // Намерен е път.
    if(x == 7 && y == 7) {
        printWay(crrWay,l+1);
        return;
    }

    // Клетката е непроходима.
    if (!labyrinth[x][y]) {
        return;
    }

    // Клетката е проходима. С цел предотвратяване на зацикляне, тази клетка се маркира като непроходима.
    labyrinth[x][y] = 0;

    // Търсене на всички пътища от четирите съседни на (x, y) клетки до клетка (7, 7)
    way(x+1, y, crrWay, l+1);
    way(x, y+1, crrWay, l+1);
    way(x-1, y, crrWay, l+1);
    way(x, y-1, crrWay, l+1);

    // "връщане назад"
    labyrinth[x][y] = 1;
}
```

Задача

Да се напише програма, която въвежда от клавиатурата без грешка булев израз от вида:

- <булев израз> ::= t | f | <операция>(<операнди>)
- <операция> ::= n | a | o
- <операнди> ::= <операнд> | <операнди>
- <операнд> ::= <булев израз>

t, f са истина и лъжа, n има само един операнд,

a, o – могат да имат повече от 1 операнд и означават съответно логическо отрицание, конюнкция и дизюнкция.

Програмата да намира и извежда стойността на въведения израз.

Задача

```
#include <iostream>
using namespace std;

bool expression();

int main() {
    cout << expression() << "\n";
    return 0;
}
```

Задача

```
bool expression () {
  char c;
  bool x, y;

  cin >> c;// c e t или f или n, a или o
  if (c == 't') {
    return true;
  }

  if (c == 'f') {
    return false;
  }

  // <израз> ::= <операция>(<операнди>)
  switch (c) {
    case 'n':
        cin >> c;                // прескачане на '('
        x = expression();
        cin >> c;                // прескачане на ')'
        return !x;
```

Задача

```
    case 'a':
      cin >> c;                 // прескачане на '('
      x = expression();
      cin >> c;                 // ',' или ')'
      while(c == ',') {
        y = expression();
        x = x && y;
        cin >> c;               // прескачане на ',' или ')'
      }
      return x;

    case 'o':
      cin >> c;                 // прескачане на '('
      x = expression();
      cin >> c;                 // ',' или ')'
      while(c == ',') {
        y = expression();
        x = x || y;
        cin >> c;               // прескачане на ',' или ')'
      }
      return x;

    default:
      cout << "Error! \n";
      return false;
    }
}
```

Задача низове

Задача 0. (0.5 т.) Да се напише програма, която въвежда низ от малки букви и извежда всички букви в него като главни (All-Caps). Низът е с максимална дължина 100 символа.

Пример:

Вход: I love cats and dogs!
Изход: I LOVE CATS AND DOGS!

Задача низове

- ```
  int diff = 'a' - 'A';
  ```

- ```
  char a = 'b' + diff;
  cout << a;
  ```

Рекурсия

Задача 3. (1.25 т.) Да се напише програма, която извежда в лексикографски ред на екрана всички комбинации от K буквени думи над азбука, съдържаща N главни букви от латинската азбука. В думите не трябва да има повтарящи се букви.
Извеждането да се реализира чрез рекурсивна функция.

Валидацията на входа не е гарантирана.

Пример: Нека N=4, K=2
Възможни думи над азбуката са: AB, AC, AD, BA, BC, BD, CA, CB, CD, DA, DB, DC

Рекурсия

```
void print(int n, int k, char *niz) {
  if (k == strlen(niz)) {
    cout << niz << endl;
    return;
  }

  int len = strlen(niz);
  for(int j = 0; j < n; j++) {
    char ch = (char)((int)'a' + j);
    if (!strchr(niz, ch)) {
      niz[len]=ch;
      print(n, k, niz);
    }
  }

  niz[len] = 0;
}
```

Рекурсия

```
int main() {
   char niz[27] = {0};
   print(4, 2, niz);

   return 0;
}
```

Какво ще изведе

```
#include <iostream>
using namespace std;

void func() {
  return;
}

int main() {
  int a = 1;
  int b = 2;
  int c = 3;
  func();
  cout << "main: a = " << a << endl;
  cout << "main: b = " << b << endl;
  cout << "main: c = " << c << endl;
  return 0;
}
```

main: a = 1
main: b = 2
main: c = 3

Какво ще изведе

```
#include <iostream>
using namespace std;

void func(int a, int b, int c) {
  cout << "func: a = " << a << endl;
  cout << "func: b = " << b << endl;
  cout << "func: c = " << c << endl;
  return;
}

int main() {
  int a = 1;
  int b = 2;
  int c = 3;
  func(a, b, c);
  cout << "main: a = " << a << endl;
  cout << "main: b = " << b << endl;
  cout << "main: c = " << c << endl;
  return 0;
}
```

func: a = 1
func: b = 2
func: c = 3

main: a = 1
main: b = 2
main: c = 3

Какво ще изведе

```
#include <iostream>
using namespace std;

void func(int a, int b, int c) {
    a = a + b;
    b = b + c;
    c = c + a;
    cout << "func: a = " << a << endl;
    cout << "func: b = " << b << endl;
    cout << "func: c = " << c << endl;
    return;
}

int main() {
    int a = 1;
    int b = 2;
    int c = 3;
    func(4, b, c);
    func(b, 5, c);
    func(b, c, 6);
    cout << "main: a = " << a << endl;
    cout << "main: b = " << b << endl;
    cout << "main: c = " << c << endl;
    return 0;
}
```

func: a = 6
func: b = 5
func: c = 9

func: a = 7
func: b = 8
func: c = 10

func: a = 5
func: b = 9
func: c = 11

main: a = 1
main: b = 2
main: c = 3

Какво ще изведе

```
#include <iostream>
using namespace std;

void func(int &a, int &b, int &c) {
    int x;
    x = a;
    a = b;
    b = x;
    c = 5;
    cout << "func: a = " << a << endl;
    cout << "func: b = " << b << endl;
    cout << "func: c = " << c << endl;
    return;
}

int main() {
    int a = 1;
    int b = 2;
    int c = 3;
    cout << "main: a = " << a << endl;
    cout << "main: b = " << b << endl;
    cout << "main: c = " << c << endl;
    func(a, b, c);
    func(b, c, a);
    func(c, a, b);
    cout << "main: a = " << a << endl;
    cout << "main: b = " << b << endl;
    cout << "main: c = " << c << endl;
    return 0;
}
```

main: a = 1
main: b = 2
main: c = 3

func: a = 2
func: b = 1
func: c = 5

func: a = 5
func: b = 1
func: c = 5

func: a = 5
func: b = 1
func: c = 5

main: a = 1
main: b = 5
main: c = 5

Какво ще изведе

```
#include <iostream>
using namespace std;

void func(int &a, int b, int& c) {
  int x;
  x = a;
  a = 4;
  c = b;
  b = x;
  cout << "func: a = " << a << endl;
  cout << "func: b = " << b << endl;
  cout << "func: c = " << c << endl;
  return;
}

int main() {
  int a = 1;
  int b = 2;
  int c = 3;
  cout << "main: a = " << a << endl;
  cout << "main: b = " << b << endl;
  cout << "main: c = " << c << endl;
  func(a, b, c);
  func(b, c, a);
  func(c, a, b);
  cout << "main: a = " << a << endl;
  cout << "main: b = " << b << endl;
  cout << "main: c = " << c << endl;
  return 0;
}
```

main: a = 1
main: b = 2
main: c = 3

func: a = 4
func: b = 1
func: c = 2

func: a = 4
func: b = 2
func: c = 2

func: a = 4
func: b = 2
func: c = 2

main: a = 2
main: b = 2
main: c = 4

Какво ще изведе

```
#include <iostream>
using namespace std;

void func(int* a, int* b, int* c) {
    int x;
    x = *a;
    *a = *b;
    *b = x;
    c = b;
    cout << "func: *a = " << *a << endl;
    cout << "func: *b = " << *b << endl;
    cout << "func: *c = " << *c << endl;
    return;
}

int main() {
    int a = 1;
    int b = 2;
    int c = 3;
    cout << "main: a = " << a << endl;
    cout << "main: b = " << b << endl;
    cout << "main: c = " << c << endl;
    func(&a, &b, &c);
    func(&b, &c, &a);
    func(&c, &a, &b);
    cout << "main: a = " << a << endl;
    cout << "main: b = " << b << endl;
    cout << "main: c = " << c << endl;
    return 0;
}
```

main: a = 1
main: b = 2
main: c = 3

func: *a = 2
func: *b = 1
func: *c = 1

func: *a = 3
func: *b = 1
func: *c = 1

func: *a = 2
func: *b = 1
func: *c = 1

main: a = 1
main: b = 3
main: c = 2

Какво ще изведе

```
#include <iostream>
using namespace std;

void func(int &x, int y, int z) {
    int a = 3;
    int b = 4;
    x = z + a;
    y = z + x;
    z = b;
    cout << "func: x = " << x << endl;
    cout << "func: y = " << y << endl;
    cout << "func: z = " << z << endl;
    cout << "func: a = " << a << endl;
    cout << "func: b = " << b << endl;
    return;
}

int main() {
    int a = 1;
    int b = 2;
    func(b, a, b);
    cout << "main: a = " << a << endl;
    cout << "main: b = " << b << endl;
    return 0;
}
```

func: a = 5
func: y = 7
func: z = 4
func: a = 3
func: b = 4

main: a = 1
main: b = 5

Какво ще изведе

```
#include <iostream>
using namespace std;

void func(const int x, int &y, int* z) {
    int a = 3;
    int b = 4;
    a = b + y;
    y = x + a;
    z = &b;
    cout << "func: x =  " << x << endl;
    cout << "func: y =  " << y << endl;
    cout << "func: *z = " << *z << endl;
    cout << "func: a =  " << a << endl;
    cout << "func: b =  " << b << endl;
    return;
}

int main() {
    int a = 1;
    int b = 2;
    func(b, a, &a);
    cout << "main: a = " << a << endl;
    cout << "main: b = " << b << endl;
    return 0;
}
```

func: a = 2
func: y = 7
func: *z = 4
func: a = 5
func: b = 4

main: a = 7
main: b = 2

Какво ще изведе

```
#include <iostream>
using namespace std;
const int a = 7;

void func(const int& x, const int* y = &a, int z = 6) {
    int a = 4;
    int b = 5;
    z = x + *y + a + b;
    cout << "func: x = " << x << endl;
    cout << "func: *y = " << *y << endl;
    cout << "func: z = " << z << endl;
    cout << "func: a = " << a << endl;
    cout << "func: b = " << b << endl;
    return;
}

int main() {
    int a = 1;
    int b = 2;
    int c = 3;
    func(a, &b, c);
    func(b, &c);
    func(c);
    cout << "main: a = " << a << endl;
    cout << "main: b = " << b << endl;
    cout << "main: c = " << c << endl;
    return 0;
}
```

func: x = 1
func: *y = 2
func: z = 12
func: a = 4
func: b = 5

func: x = 2
func: *y = 3
func: z = 14
func: a = 4
func: b = 5

func: x = 3
func: *y = 7
func: z = 19
func: a = 4
func: b = 5

main: a = 1
main: b = 2
main: c = 3

Какво ще изведе

Да се дефинира функция, която разделя символен низ на части по зададени разделители ‘ ’, ‘;’, ‘-’, ‘,’.

```
// Връща указател към първото срещане на разделител
// след символ неразделител. Ако няма такова срещане -
// връща NULL

// Преброява частите, на които ще бъде разделен низът

// Извършва разделянето.
// Връща масив от частите, на които е разделен низът.
// *Заделя памет за самите части и за масива от тях
```

```
#include <iostream>
#include <string.h>
using namespace std;

const int MAX_LEN = 100;

// Връща указател към първото срещане на разделител
// след символ неразделител. Ако няма такова срещане -
// връща NULL
const char* nextToken(const char* text,
                      const char* delimeters) {
  while(*text && strchr(delimeters, *text)) {
    ++text;
  }

  if(!*text) {
    return NULL;
  }

  while(!(strchr(delimeters, *text))) {
    ++text;
  }

  return text;
}
```

```
// Преброява частите, на които ще бъде разделен низът
int tokenCnt(const char* text, const char* delimeters) {
    int cnt = 0;
    const char* p = text;
    while(p = nextToken(p, delimeters)) {
      ++cnt;
    }
    return cnt;
}
```

```
// Извършва разделянето.
// Връща масив от частите, на които е разделен низът.
// Заделя памет за самите части и за масива от тях
char** tokenize(const char* text, const char* delimeters,
                char**& tokens, int& num) {
  num = tokenCnt(text, delimeters);
  tokens = new char*[num];
  const char* start = text;
  const char* end;
  for(int i = 0; i < num; ++i) {
    end = nextToken(start, delimeters);
    if (end) {
      while(*text && strchr(delimeters, *text)) {
        ++text;
      }
      int len = (int)(end - start);
      tokens[i] = new char[len + 1];
      strncpy(tokens[i], start, len);
      tokens[i][len] = '\0';
      start = end;
    }
  }
  return tokens;
}
```

```
int main() {
    char text[MAX_LEN];
    char* delims = ",; -";
    int num;
    char** tokens;
    cout << "Въведете текст ";
    cin.getline(text, MAX_LEN);
    tokenize(text, delims, tokens, num);
    for(int i = 0; i < num; ++i) {
      cout << tokens[i] << endl;     // извеждане
      delete tokens[i];              // и освобождаване на паметта
    }
    delete tokens;                 // освобождаване на паметта за масива
    return 0;
}
```

Какво ще изведе

Два стълба на една матрица си приличат, ако съвпадат множествата от числата, съставящи стълбовете. Да се напише програма, която установява дали съществуват два стълба на квадратната матрица A с размерност n, които си приличат.

1 <= n <=20

```
#include <iostream>
#include <iomanip>
using namespace std;
const int MAX_SIZE = 20;

int main() {
  cout << "Въведете стойност на n в интервала "
       "[1, " << MAX_SIZE << "]: ";
  int n;
  cin >> n;
  int a[MAX_SIZE][MAX_SIZE];

  // въвеждане на матрицата A
  int i, j;
  cout << "Въведете " << n << " реда с по "
       << n << " цели числа.\n";
  for(i = 0; i < n; i++) {
    for(j = 0; j < n; j++) {
      cin >> a[i][j];
    }
  }
```

Функция!!!

Какво ще изведе

```
// извеждане на матрицата
for(i = 0; i < n; i++) {
  for(j = 0; j < n; j++) {
    cout << setw(5) << a[i][j];
  }
  cout << endl;
}
```

Функция!!!

```
// проверка дали съществуват два стълба, които си приличат
bool flag = false;
int p, q;
for(p = 0; p < n-1 && !flag; p++) {
  for(q = p+1; q < n && !flag; q++) {
  // проверка дали p-ти стълб прилича на q-ти стълб
  // проверка дали всеки елемент на p-ти стълб
  // принадлежи на q-ти стълб на A
  flag = true;
  i = 0;
  int j;
  while(i < n && flag) {
    j = -1;
    do {
      j++;
    } while(a[j][q] != a[i][p] && j < n-1);

    if (a[j][q] != a[i][p]) {
      flag = false;
    }
    i++;
  }
```

Функция!!!

Функция!!!

Какво ще изведе

```
if (flag) {
  // проверка дали всеки елемент на q-ти стълб
  // принадлежи на p-ти стълб на A
  j = 0;
  while(j < n && flag) {
    int i = -1;
    do {
      i++;
    } while(a[i][p] != a[j][q] && i < n-1);

    if (a[i][p] != a[j][q]) {
      flag = false;
    }
    j++;
  }
}

}
}
```

Същата Функция!!!

Какво ще изведе

```
  if(flag) {
    cout << "Съществуват 2 стълба, които си приличат.\n";
  } else {
    cout << "Не съществуват 2 стълба, които си приличат.\n";
  }
  return 0;
}
```

Задачи

Дадени са два символни низа s1 и s2, съставени от малки латински букви. Да се напише програма, която проверява дали съществува функция, изобразяваща s1 в s2.

Задачи

```
#include <iostream>
#include <string.h>
using namespace std;
const int MAX_SIZE = 20;

int main() {
    char s1[MAX_SIZE], s2[MAX_SIZE];
    // масивът function показва всяка от намерените
    // до момента стойности на функцията. Използва
    // се 0 за обозначаване на неопределена стойност
    char function[26] = {0};
    cout << "s1= ";
    cin.getline(s1, MAX_SIZE);
    cout << "s2= ";
    cin.getline(s2, MAX_SIZE);

    // дължина и брояч
    unsigned int l = strlen(s1);
    unsigned int i = 0;
```

Задачи

```
    if(l > strlen(s2)) {
        cout << "Свойството не е в сила.\n";
    } else {
       // изпълнението завършва, когато низовете се изчерпат
       // или е намерена двузначност на изображението
       while(i < l && (function[s1[i]-'a'] == 0 ||
                       function[s1[i]-'a'] == s2[i])) {
         function[s1[i]-'a'] = s2[i];
         i++;
       }
       if(i == l) {
         cout << "Свойството е в сила.\n";
       } else {
         cout << "Свойството не е в сила.\n";
       }
   }
   return 0;
}
```

Рекурсия

Задача 1. (1 т.) Да се напише функция **mySort**, която сортира възходящо елементите на масив от цели числа. За сравнение на елементите на масива да се използва подадена като параметър произволна функция **comparator**. Функцията **mySort** приема като аргументи указател към масив от цели числа, размер на масива и указател към функцията **comparator**.
Функцията **comparator** приема 2 аргумента (цели числа) и връща стойност **true**, ако първият аргумент е по-голям от втория по някакъв критерий и **false** в противен случай.

Рекурсия

Задача 2. (1.25 т.) Да се напише програма, която въвежда кодирано изречение (низ) и го декодира по следния начин:

- Думите на четни позиции (вкл. 0) са първата част на декодираното изречение и са изписани правилно.

- Думите на нечетни позиции са втората част на декодираното, записани наобратно и в обратен ред.

Низът е с максимална дължина 100 символа.

Пример:

Вход: We samtsirhC wish yrraM you a

Изход: We wish you a Merry Christmas

```
cout << “Край”;
```