**Лабораторная работа № 10. Рекурсивные алгоритмы**

|  |  |  |
| --- | --- | --- |
| 1. Изучить использование простых *рекурсивных* функций на примере программы, представленной в правой части. Определить признак конца ввода чисел, проанализировав текст программы.  Добавить операторы, позволяющие определить, сколько раз функция **binPrn** вызывает сама себя. | #include <stdio.h>  int count = 0;  void binPrn(unsigned num)  {  count++;  if (num / 2)  binPrn(num / 2);  putchar(num % 2 + '0');  }  int main(void)  {  int c;  while (1)  {  printf("Number: ");  if (scanf\_s("%d", &c) != 1 || !c)  break;  binPrn(c);  printf("\nNumber of recursive calls: %d\n", count);  }  return 0;  } |  |
| 2. Выполнить программу, разработанную с использованием рекурсии.  Реализовать алгоритм без использования рекурсии. | #include <iostream>  char s[100];  bool isPalindrom(char s[100]);  void main()  {  setlocale(LC\_CTYPE, "Russian");  printf("\nВведите строку: ");  gets\_s(s);  if (isPalindrom(s))  printf("Строка - палиндром");  else  printf("Строка - не палиндром");  }  bool isPalindrom(char s[100])  {  bool l;  char s1[100];  if (strlen(s) <= 1)  return true;  else  {  l = s[0] == s[strlen(s) - 1];  strncpy\_s(s1, s + 1, strlen(s) - 2);  s1[strlen(s) - 2] = '\0';  return l && isPalindrom(s1);  }  }  Реализация без рекурсии  #include <iostream>  #include <cstring>  #include <cctype>  bool isPalindrom(char s[100]);  int main()  {  setlocale(LC\_CTYPE, "Russian");  char s[100];  printf("\nВведите строку: ");  gets\_s(s);  if (isPalindrom(s))  printf("Строка - палиндром");  else  printf("Строка - не палиндром");  return 0;  }  bool isPalindrom(char s[100])  {  int length = strlen(s);  if (length <= 1)  return true;  for (int i = 0; i < length / 2; i++)  {  if (s[i] != s[length - 1 - i])  return false;  }  return true;  } |  |
| 3. Выполнить программу, приведенную в правой части. Записать ее условие.  Добавить любое слово в массив **w** и проанализировать результат работы программы.  Если некоторое слово в массиве **w** не подходит для решения задачи, то добавить операторы вывода сообщения о том, что его нужно изменить. | #include <iostream>  #include <cstring>  using namespace std;  const char\* w[] = { "ПАЛКА", "ЛЯГУШКА", "КАПЛЯ", "КАРТА", NULL };  bool test(const char\* s, const char\* r) {  int n = strlen(s);  if (n == 0) {  return true;  }  for (; \*s != '\0' && n > 1; s++, n--) {  if (strncmp(s, r, n) == 0) {  return true;  }  }  return false;  }  int step(const char\* lw) {  setlocale(LC\_CTYPE, "Russian");  int n;  for (n = 0; w[n] != NULL; n++) {  if (\*w[n] != '\0') {  break;  }  }  if (w[n] == NULL) return 1;  for (n = 0; w[n] != NULL; n++) {  const char\* pw;  if (\*w[n] == '\0') {  continue;  }  pw = w[n];  w[n] = "";  if (test(lw, pw)) {  if (step(pw)) {  cout << pw << "\n";  return 1;  }  }  w[n] = pw;  }  return 0;  }  int main() {  step("");  return 0;  } | Программа представляет собой рекурсивный алгоритм, который пытается построить цепочку слов из массива w, где каждое следующее слово начинается с последовательности букв, которая является окончанием предыдущего слова (с длиной от 2 символов и более). |
|  | | |
| #include <iostream>  using namespace std;  int F(int m, int n) {  if (m == 0 || n == 0)  return n + 1;  return F(m - 1, F(m, n - 1));  }  double Var\_12(double x, int n) {  if (n == 1)  return cos(x);  return cos(n \* x) + Var\_12(x, n - 1);  }  int main() {  setlocale(LC\_ALL, "rus");  cout << "Введите номер варианта:";  int choice;  cin >> choice;  switch (choice) {  case 10: {  cout << "Введите целые, неотрицательные n и m\n";  int n, m;  cin >> n >> m;  cout << "F = " << F(m, n);  break;  }  case 12: {  cout << "Введите целое число n: ";  int n;  cin >> n;  cout << "Введите число x:";  double x;  cin >> x;  cout << "y = " << Var\_12(x, n);  break;  }  default: break;  }  return 0;  } | | |
| 1. Ввести цифру **А**, записать в файл все возможные числа, состоящие из цифр, не превышающих или равных **A**. Количество цифр в числах должно быть равно **А**.  Примечание: использовать дополнительный массив | #include <iostream>  #include <fstream>  using namespace std;  void fVarA(int A, int len, ofstream &fout,int buf[]) {  if (len == 0) {  for (int i = A; i > 0; i--) {  fout << buf[i];  }  fout << ' ';  return;  }  for (int i = 0; i <=A; i++) {  if (i == 0 && len == A) {  i++;  }  buf[len] = i;  fVarA(A, len - 1, fout, buf);  }  return;  }  int main() {  char fname[] = "ans.txt";  int A,buf[256];  ofstream fout(fname);  cin >> A;  fVarA(A, A, fout, buf);  return 0;  } | |
| 2. Задача проведения границы на карте («создание военных блоков»). Страны на карте заданы матрицей смежности. Если страны **i**, **j** имеют на карте общую границу, то элемент матрицы **A[i, j]** равен 1, иначе 0.  Необходимо разбить страны на две группы так, чтобы количество пар смежных стран из противоположных групп было минимальным. | #include <iostream>  using namespace std;  void backtrack(int\*\* adj, int n, int current, int\* groups, int\* best\_groups, int& min\_cut, int current\_cut = 0) {  if (current == n) {    bool has\_cut = false;  for (int i = 0; i < n; ++i) {  for (int j = i + 1; j < n; ++j) {  if (adj[i][j] && groups[i] != groups[j]) {  has\_cut = true;  break;  }  }  if (has\_cut) break;  }  if (has\_cut && current\_cut < min\_cut) {  min\_cut = current\_cut;  for (int i = 0; i < n; ++i) {  best\_groups[i] = groups[i];  }  }  return;  }    groups[current] = 0;  int new\_cut = current\_cut;  for (int i = 0; i < current; ++i) {  if (adj[current][i] && groups[i] != groups[current]) {  new\_cut++;  }  }  if (new\_cut < min\_cut) {  backtrack(adj, n, current + 1, groups, best\_groups, min\_cut, new\_cut);  }    groups[current] = 1;  new\_cut = current\_cut;  for (int i = 0; i < current; ++i) {  if (adj[current][i] && groups[i] != groups[current]) {  new\_cut++;  }  }  if (new\_cut < min\_cut) {  backtrack(adj, n, current + 1, groups, best\_groups, min\_cut, new\_cut);  }  }  int minCutPartition(int\*\* adj, int n, int\* best\_groups) {  if (n == 0) return 0;  int\* groups = new int[n];  for (int i = 0; i < n; ++i) groups[i] = -1;  int min\_cut = INT\_MAX;  backtrack(adj, n, 0, groups, best\_groups, min\_cut);  delete[] groups;  return min\_cut;  }  int main() {  setlocale(LC\_ALL, "rus");  const int n = 4;    int\*\* adj = new int\* [n];  for (int i = 0; i < n; ++i) {  adj[i] = new int[n];  for (int j = 0; j < n; ++j) {  adj[i][j] = 0;  }  }    adj[0][1] = adj[1][0] = 1; // Страны 0 и 1 имеют общую границу  adj[0][2] = adj[2][0] = 1; // Страны 0 и 2 имеют общую границу  adj[1][2] = adj[2][1] = 1; // Страны 1 и 2 имеют общую границу  adj[1][3] = adj[3][1] = 1; // Страны 1 и 3 имеют общую границу  adj[2][3] = adj[3][2] = 1; // Страны 2 и 3 имеют общую границу  int\* best\_groups = new int[n];  int result = minCutPartition(adj, n, best\_groups);  cout << "Минимальные смежные границы между группами: " << result << endl;  cout << "Распределение груп: ";  for (int i = 0; i < n; ++i) {  cout << best\_groups[i] << " ";  }  cout << endl;  for (int i = 0; i < n; ++i) {  delete[] adj[i];  }  delete[] adj;  delete[] best\_groups;  return 0;  } | |
| 3. Дано **n** различных натуральных чисел (**n = 5**). Напечатать все перестановки этих чисел. | #include <iostream>  #include <fstream>  using namespace std;  void fVarA(int n, int index, int buf[],int ans[]) {  if (index == 0) {  for (int i = 0; i < n; i++) {  cout << ans[i] << ' ';  }  cout << '\n';  return;  }  int newbuf[256];  for (int i = 0; i < n; i++) {  newbuf[i] = buf[i];  }  for (int i = 0; i < n; i++) {  if (newbuf[i] != -1) {  int temp = newbuf[i];  ans[index-1] = temp;  newbuf[i] = -1;  fVarA(n, index - 1, newbuf,ans);  newbuf[i] = temp;  }  }  return;  }  int main() {  setlocale(LC\_ALL, "rus");  int n, ans[256];  cout << "Введите количество элементов: ";  cin >> n;  cout << "Элементы массива: ";  int\* buf = new int[n];  for (int i = 0; i < n; i++) {  cin >> buf[i];  }  fVarA(n, n, buf,ans);  return 0;  } | |