Министерство образования и науки Российской Федерации

Волжский политехнический институт (филиал) федерального государственного бюджетного образовательного учреждения высшего профессионального образования «Волгоградский государственный технический университет»
(ВПИ (филиал) ВолгГТУ)

Факультет «*Вечерний факультет*»

Кафедра «*Информатика и технология программирования*»

КОНТРОЛЬНАЯ РАБОТА

по дисциплине «*Программирование и основы алгоритмизации*»

на тему ***РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА С ИСПОЛЬЗОВАНИЕМ АЛГОРИТМОВ МОДУЛЬНОГО ПРОГРАММИРОВАНИЯ***

(вариант №2)

Студент *Сергей Петрович Иванов*
(имя, отчество, фамилия)

Группа *ВХАЗ-250*

Оценка _____
(зачтено/незчтено)

Проверил _____ *ст.преп., О.Ф.Абрамова*
(подпись и дата подписания) (долж., инициалы и фамилия)

Нормоконтролер _____ *Н.А. Билялова*
(подпись, дата подписания) (инициалы и фамилия)

Волжский, 2012 г.

1. ПОСТАНОВКА ЗАДАЧИ

Задача 1

Дан одномерный массив из n элементов целого типа. Определить:

- количество значений массива, равных максимальному значению;

- отсортировать по возрастанию элементы массива, начиная и заканчивая индексами, заданными пользователем (например: с 3-го по 10-й);

- сумму элементов, расположенных после первого нуля.

Задача 2

Дан двумерный массив, размерностью M x N. Определить:

- количество строк, не содержащих ни одного нулевого элемента;

- поменять местами элементы двух столбцов, номера которых задаются пользователем.

Дополнительные требования

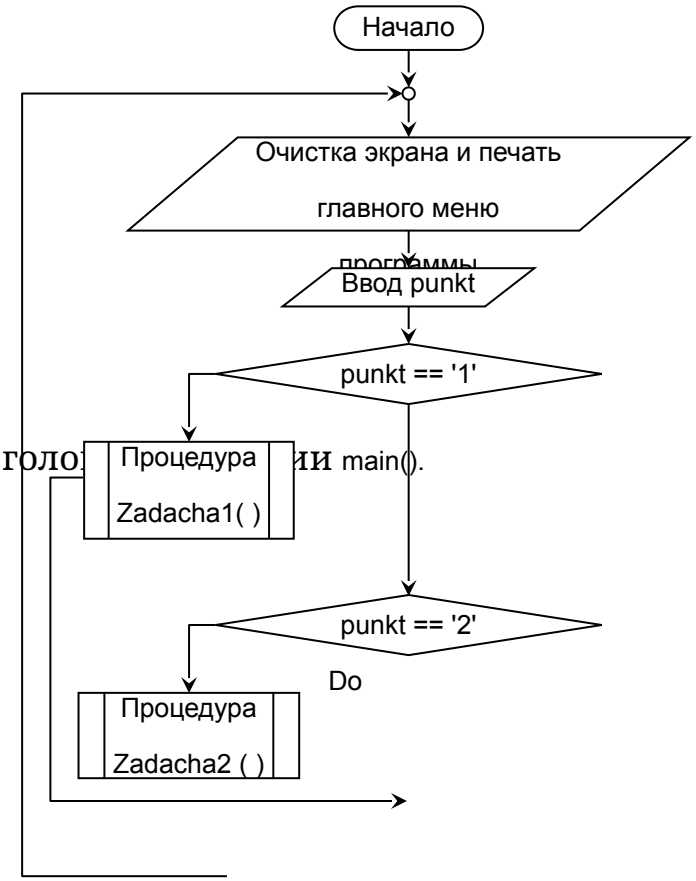
Все задачи варианта объединяются одним общим интерфейсом. Выполнение каждой задачи осуществляется как последовательный вызов всех необходимых подпрограмм, согласно разработанному алгоритму. Предусмотреть возможность повтора выполнения каждой задачи. Результаты должны печататься с максимально возможными комментариями. Размерность массивов должна задаваться пользователем программного средства.

1. Описание глобальных переменных

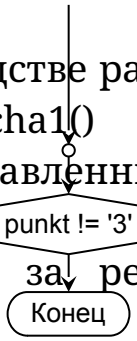
Было разработано программное средство согласно заявленным в методических указаниях требованиям. Для успешной работы данного программного средства в соответствии с предъявляемыми требованиями разработчиком было решено глобальные переменные не использовать. Это повышает мобильность программы и и соответствует основным принципам структурного программирования.

Выполнение каждой подзадачи производится в отдельной функции, вызов которых производится из главного меню, расположенного в головной функции main(). Организация работы меню выполнена с использованием одной переменной – punkt. В эту переменную считывается значение, вводимое пользователем программного средства и соответствующее тому номеру пункта меню, которое пользователь выбирает для выполнения.

Блок-схема голо[illegible]ии main().



2. Описание основных подпрограмм

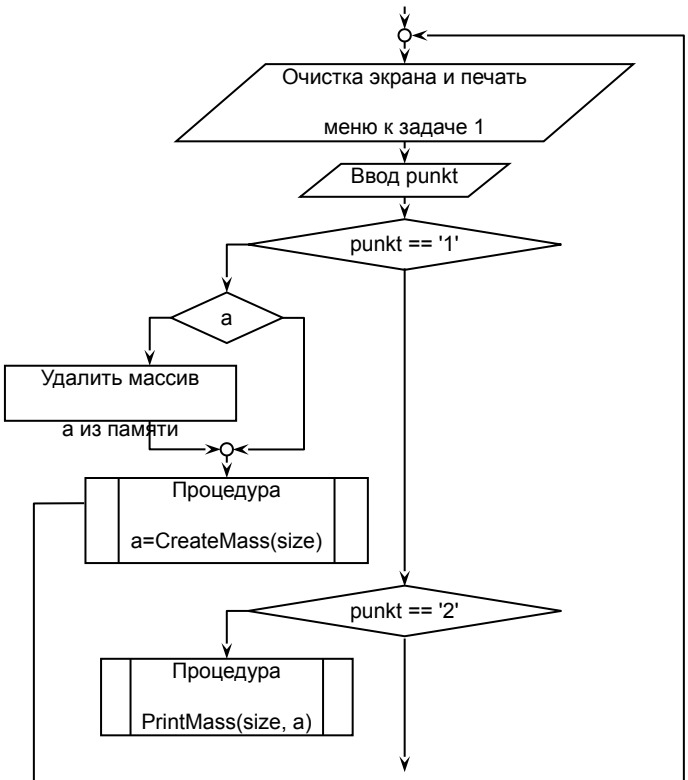


В данном программном средстве разработчиком были составлены 2 основные функции Zadacha1() и Zadacha2() соответственно, реализующие одну из поставленных в условии задач. При этом каждая из этих функций имеет в себе обращение еще к ряду подпрограмм, отвечающих за реализацию отдельных типовых алгоритмов.

Описания функций программного средства:

- Zadacha1() – функция содержит подменю, пункты которого соответствую подзадачам, изложенным в задании Задача 1 (см. «Постановка задачи»); входных параметров функция не имеет; возвращаемого значения функция не формирует.

- Zadacha2() - функция содержит подменю, пункты которого соответствую подзадачам, изложенным в задании Задача 2 (см. «Постановка задачи»); входных параметров функция не имеет; возвращаемого значения функция не формирует



Блок-схема подпрограммы Zadacha1

Do

да

да нет

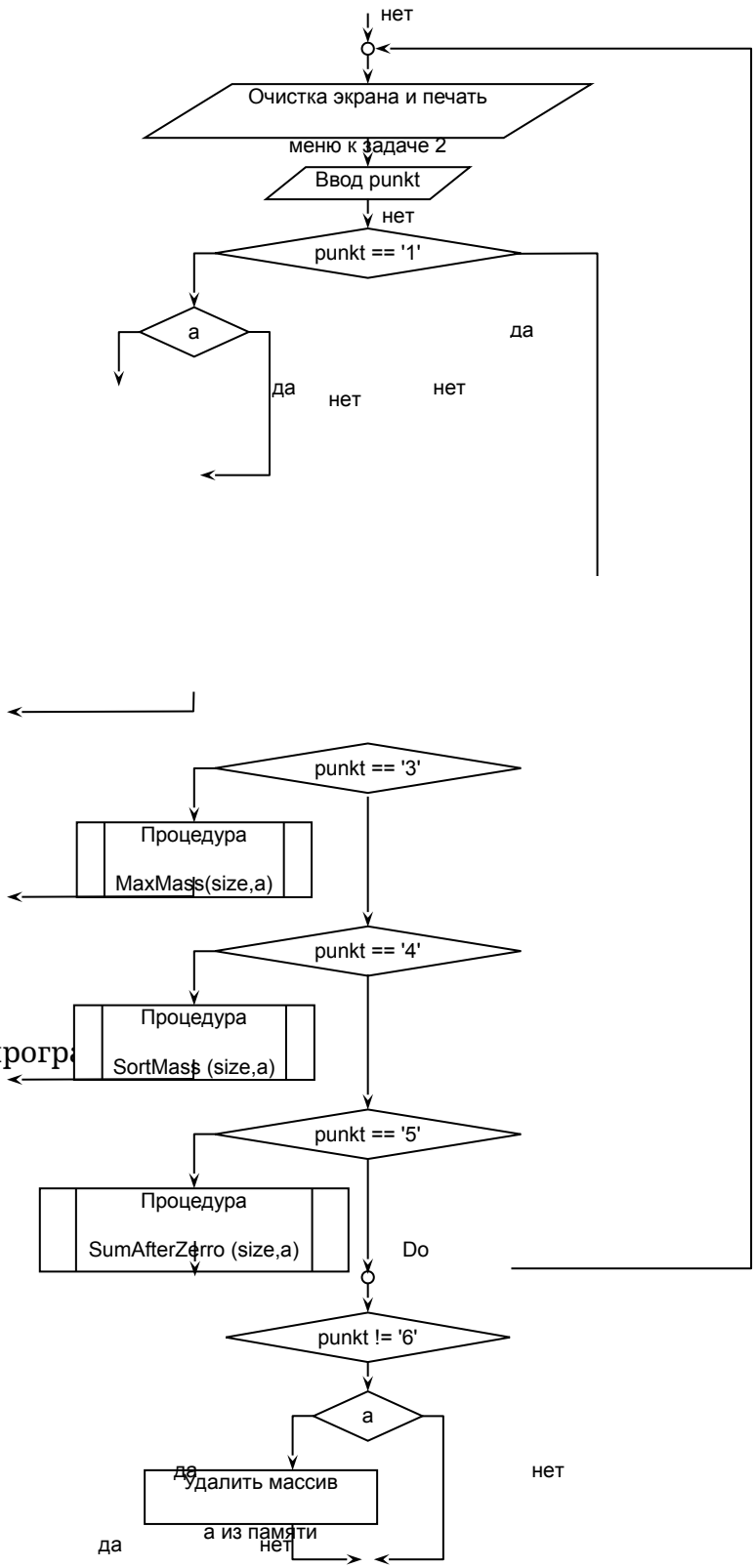
нет

да нет

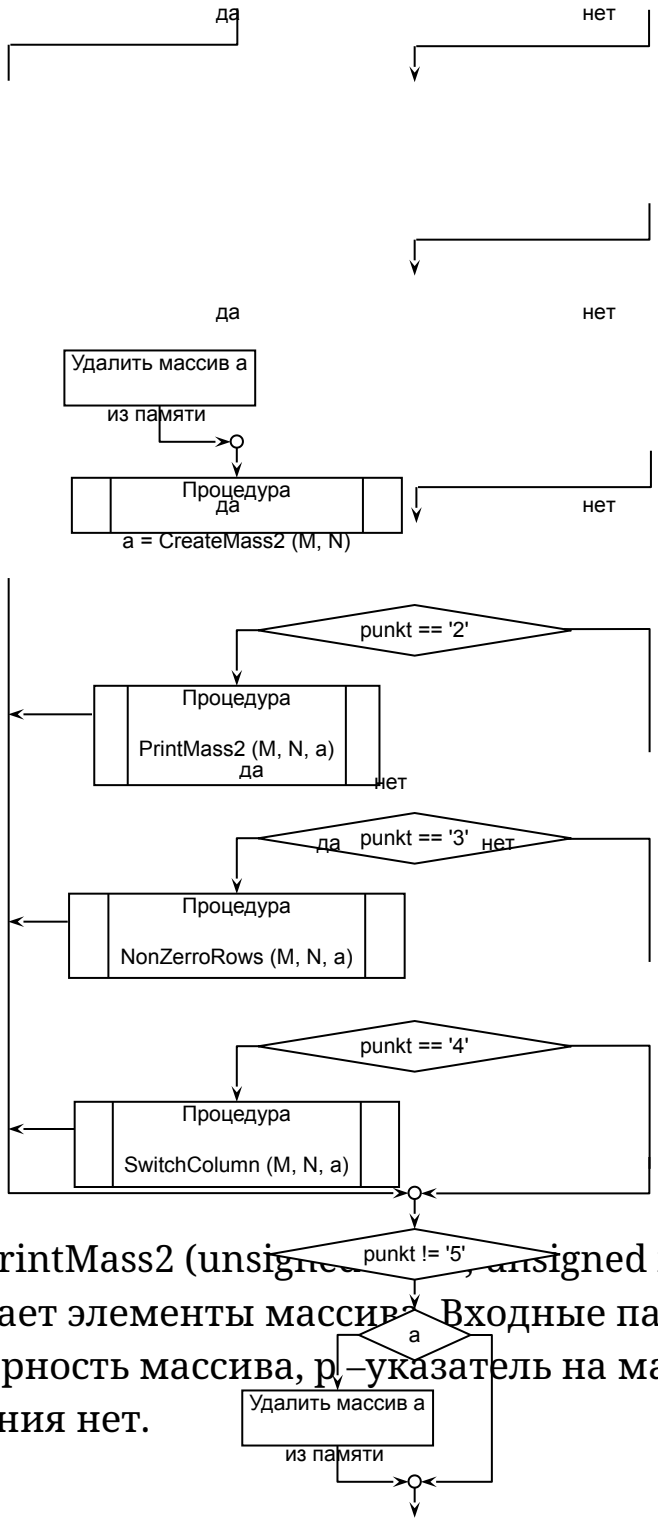
нет

да

да

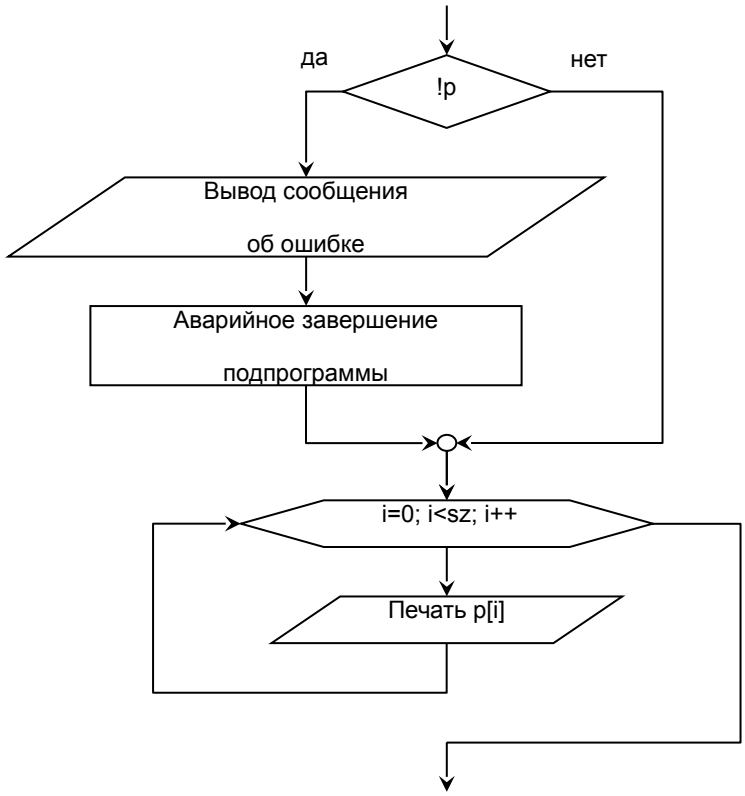


Блок-схема подпрогра



- void PrintMass2 (unsigned int M, unsigned int N, int **p) – функция печатает элементы массива. Входные параметры: N, M – размерность массива, p –указатель на массив. Выходного значения нет.

Блок-схема подпрограммы PrintMass



- Функция void **SortMass** (unsigned int sz, int* p) – функция сортирует элементы массива между заданными индексами

Входные параметры:

sz – количество элементов в массиве

p – указатель на массив

Локальные переменные:

к1 – начальный индекс элемента массива

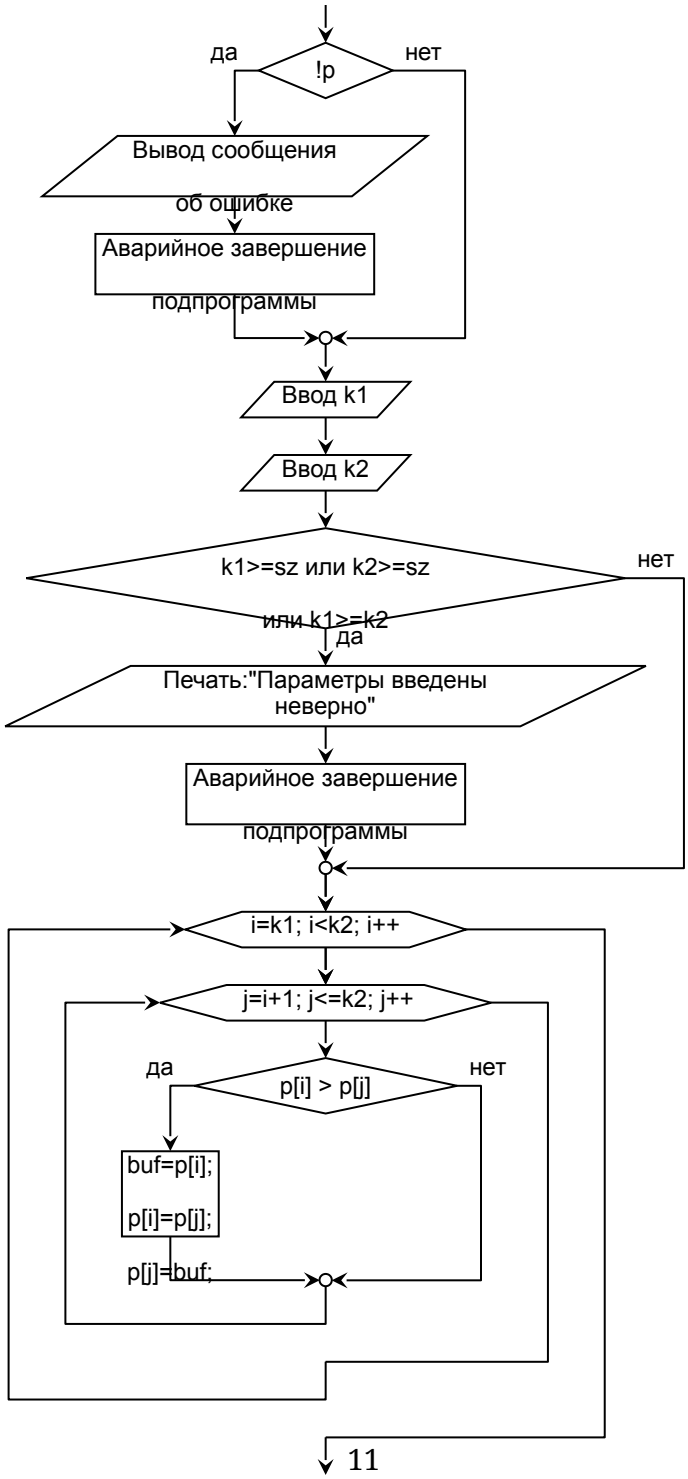
к2 – конечный индекс элемента массива

i, j – индексы текущего элемента массива

buf – дополнительная переменная для обмена значений двух элементов массива

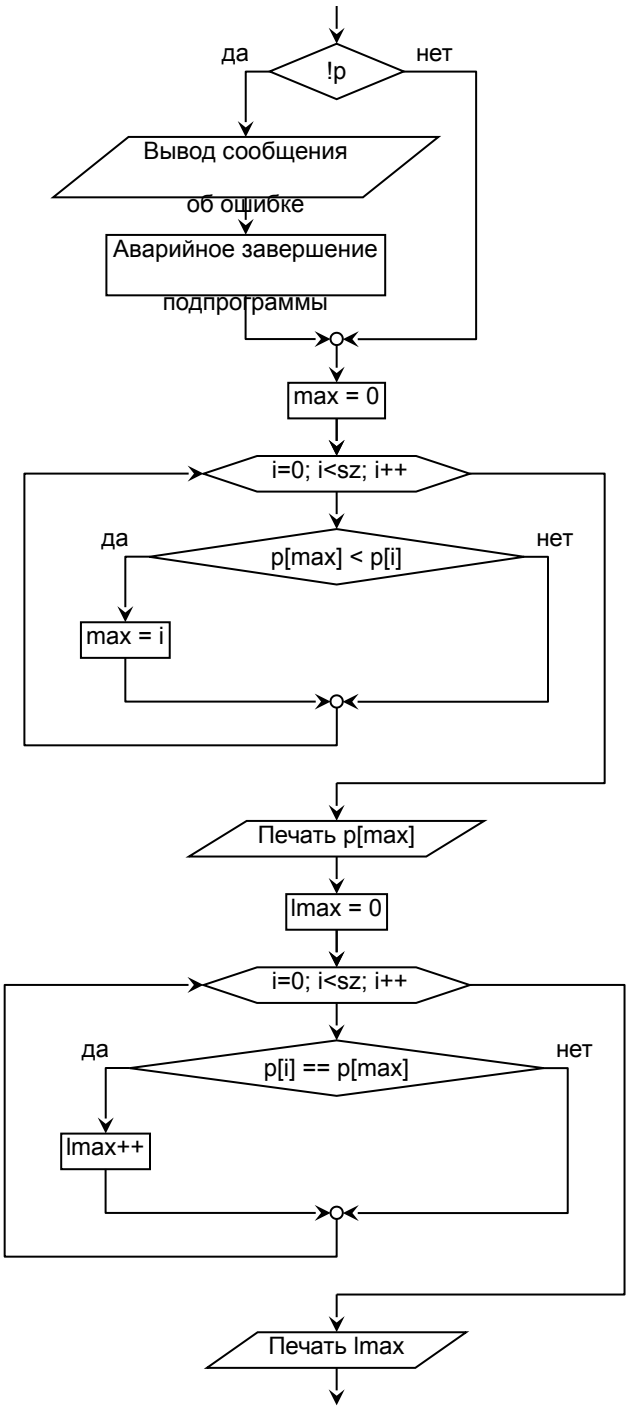
В функции предусмотрено аварийное завершение работы с выводом соответствующего сообщения при не правильном вводе пользователем значений индексов элементов.

Блок-схема подпрограммы SortMass

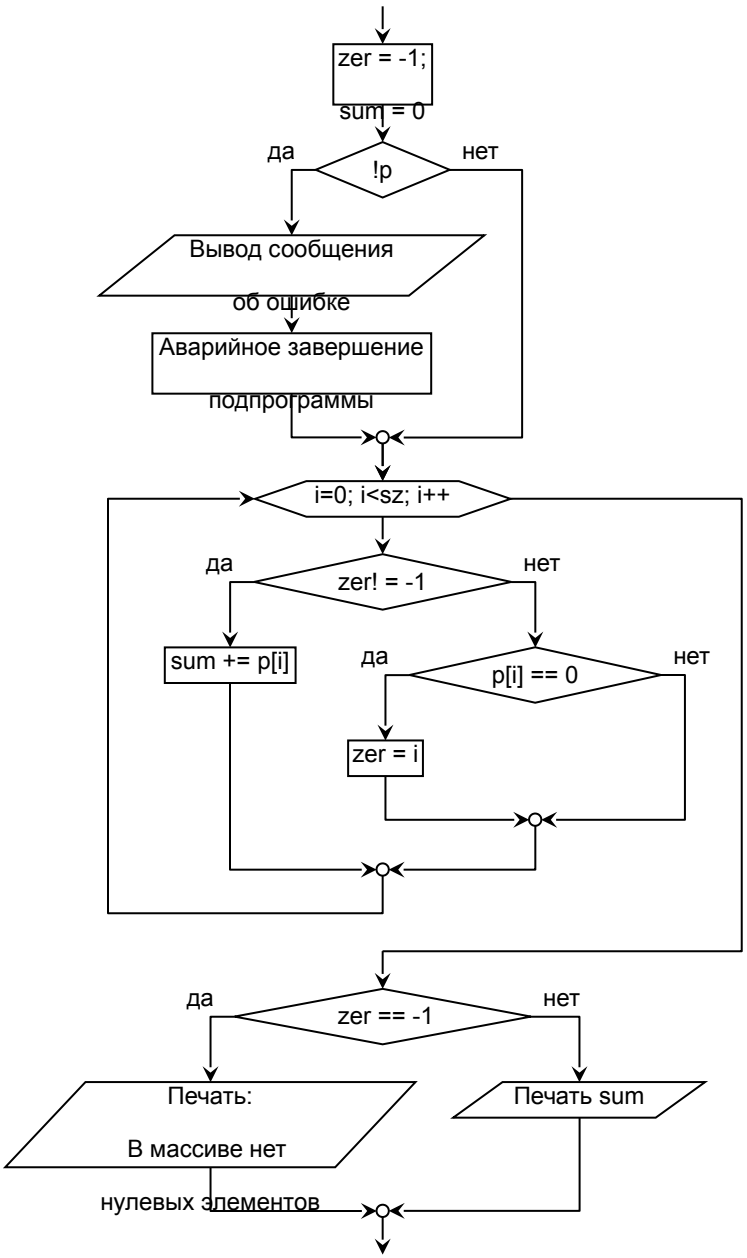


И так далее. Так как это ПРИМЕР выполнения семестровой, поэтому описания оставшихся функций выполняться не будет.

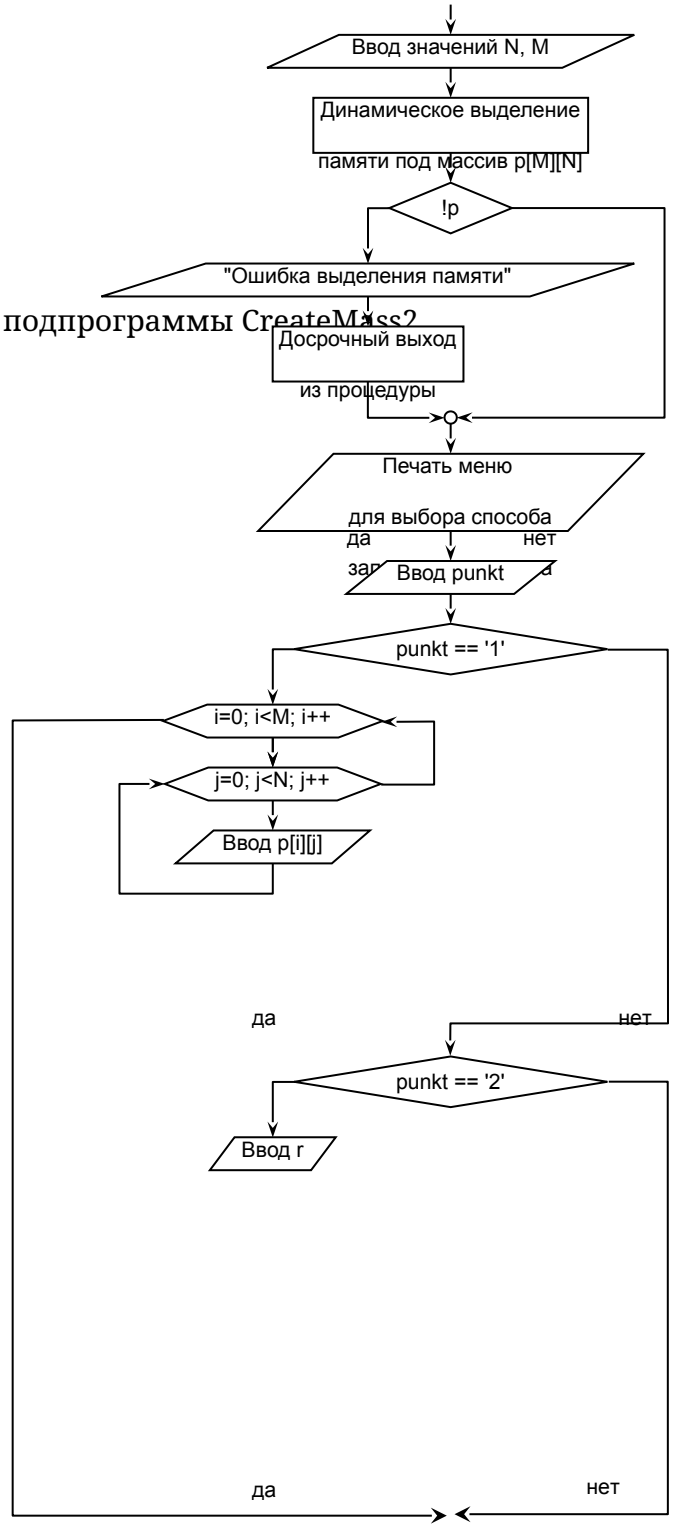
Блок-схема подпрограммы MaxMass



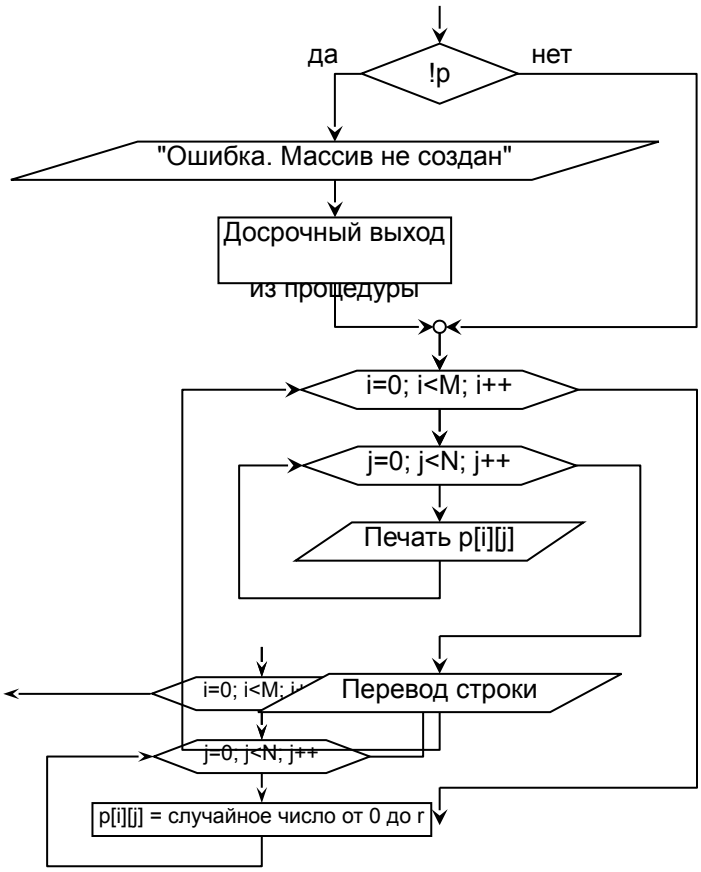
Блок-схема подпрограммы SumAfterZerro



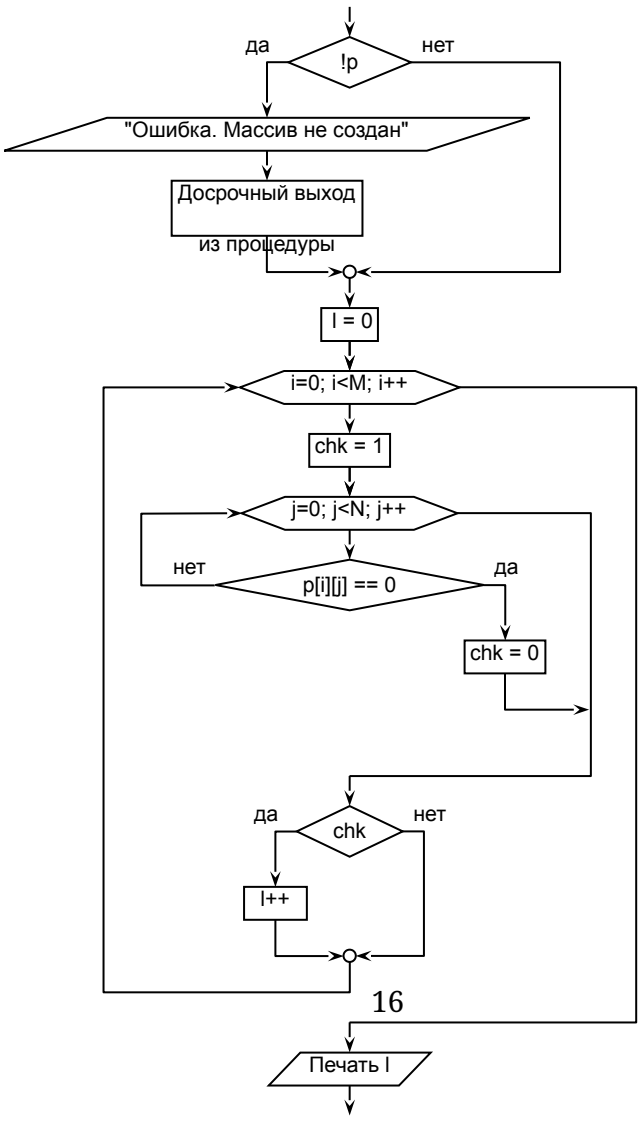
Блок-схема подпрограммы CreateMass2



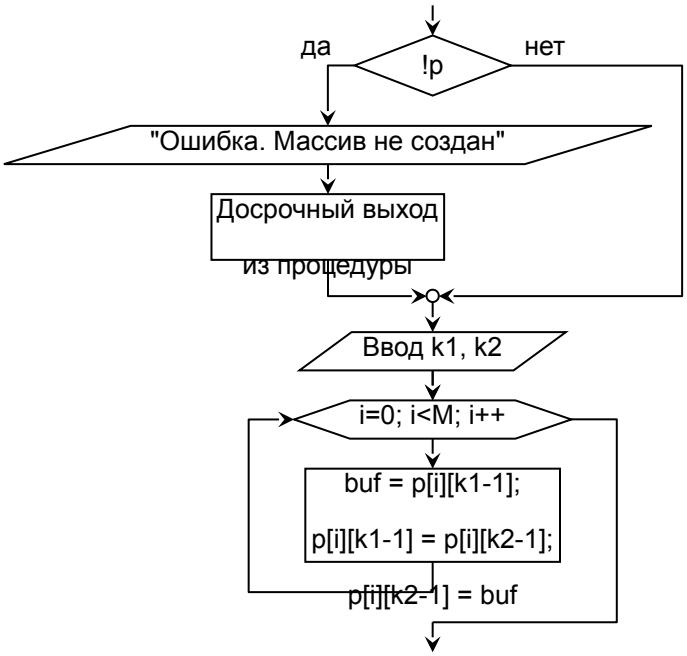
Блок-схема подпрограммы PrintMass2



Блок-схема подпрограммы NonZerroRows



Блок-схема подпрограммы SwitchColumn



3. Листинг программного продукта на языке C++

```
// Подключение библиотечных файлов C++:
#include<stdlib.h>      // необходим для операций с генератором случайных чисел
#include<iostream.h>    // необходим для операций потокового ввода-вывода
#include<conio.h>       // необходим для работы с консолью
#include<stdio.h>       // необходим для форматированного ввода-вывода


// Прототипы функций
void Zadacha1 ();
void Zadacha2 ();
int* CreateMass (unsigned int &);
int** CreateMass2 (unsigned int &, unsigned int &);
void PrintMass (unsigned int, int*);
void PrintMass2 (unsigned int, unsigned int , int**);
void SortMass (unsigned int, int*);
void MaxMass (unsigned int, int*);
void SumAfterZerro (unsigned int, int*);
```

```
void NonZerroRows (unsigned int, unsigned int, int**);
void SwitchColumn (unsigned int, unsigned int, int**);

// Главная программа
void main()
{

char punkt;

do
  {
  clrscr();
  cout << "\n   СЕМЕСТРОВАЯ РАБОТА. ВАРИАНТ 8.\n";
  cout << "\n   ГЛАВНОЕ МЕНЮ\n\n";
  cout << "1. ЗАДАЧА 1\n";
  cout << "2. ЗАДАЧА 2\n";
  cout << "3. ВЫХОД\n";

  punkt=getch();

  switch (punkt)
    {
     case '1':
      Zadacha1 ();
        break;
     case '2':
        Zadacha2 ();
      break;
    }
  }while (punkt!='3');


}









// ********************************************************************

// ****************************** ФУНКЦИИ ******************************

// ********************************************************************


void Zadacha1 ()
{

char punkt;
int *a=NULL; // указатель на создаваемый массив
unsigned int size; // размер массива

do
```

```
{
clrscr();
cout << "\n   ЗАДАЧА 1. \n   МЕНЮ\n\n";
cout << "1. Создать массив\n";
cout << "2. Распечатать массив\n";
cout << "3. Подсчет максимальных значений\n";
cout << "4. Сортировка части массива\n";
cout << "5. Расчет суммы элементов после первого нуля\n";
cout << "6. Вернуться в главное меню\n";

punkt=getch();

switch (punkt)
  {
   case '1':
     if (a) delete[] a;
     a=CreateMass(size);
     break;
   case '2':
     PrintMass(size, a);
     break;
   case '3':
     MaxMass(size,a);
     break;
   case '4':
     SortMass (size,a);
     break;
   case '5':
     SumAfterZerro (size,a);
     break;

  }

}while (punkt!='6');

if (a) delete[] a; // освобождение памяти

return;

}


void Zadacha2 ()
{
unsigned int M,N; // Размер массива
int i,j;
char punkt;
int **a=NULL;  // указатель на создаваемый массив

do{
  clrscr();
  cout << "\n   ЗАДАЧА 2. \n   МЕНЮ\n\n";
  cout << "1. Создать двумерный массив\n";
  cout << "2. Распечатать массив\n";
  cout << "3. Подсчет строк без нулевых элементов\n";
  cout << "4. Поменять местами элементы столбцов\n";
  cout << "5. Вернуться в главное меню\n";
  punkt=getch();
```

```
  switch (punkt)
    {
      case '1':
      // Удаление массива из памяти
       if (a)
      {
       for (i=0;i<M;i++)  delete[] a[i];
       delete [] a;
      }
      a=CreateMass2(M,N);
      break;
      case '2':
      PrintMass2(M,N,a);
      break;
      case '3':
        NonZerroRows (M,N,a);
        break;
      case '4':
        SwitchColumn (M,N,a);
        break;
    }


}while (punkt!='5');


// Удаление массива из памяти
if (a)
  {
   for (i=0;i<M;i++)  delete[] a[i];
   delete [] a;
  }


return;
}


//---------------------------------------------------------------------
// Распечатка двумерного массива
//---------------------------------------------------------------------
void PrintMass2 (unsigned int M, unsigned int N, int **p)
{

int i,j;

if (!p)
  {
    cout << "\n\nОшибка! Массив еще не создан.";
    getch();
    return;
  }


printf ("\n\nМассив a[%d][%d] имеет вид:\n",M,N);
for (i=0;i<M;i++)
  {
```

```
   for (j=0;j<N;j++) printf("%4d",p[i][j]);
   printf("\n");
  }

getch();

return;
}


//---------------------------------------------------------------------
// Выделение памяти под двумерный массив и заполнение его
//---------------------------------------------------------------------
int** CreateMass2 (unsigned int &M,unsigned int &N)
{
int i,j,r;
char punkt;

cout << "\n\nЗадайте размер для двумерного массива \n";
cout << "Строк в массиве:  M = "; cin >> M;
cout << "Столбцов в массиве:  N = "; cin >> N;


//Динамически определим память для двумерного массива
int **p=new int*[M]; // сначала массив из M указателей на int
// при этом переменная p является так назыв. указателем на указатель
for (i=0;i<M;i++)  p[i]=new int[N]; // теперь каждый указатель укажет на массив N
целых

if (!p)
  {
   cout << "\n\nОшибка! Невозможно создать массив! \nПопробуйте задать меньший
размер.";
   getch();
   return NULL;
  }

cout << "\n\nВвод значений в массив:";
cout << "\n1. Вручную";
cout << "\n2. Автоматически (случайные значения)\n";
punkt=getch();

switch (punkt)
  {
   case '1':
     cout << "\nВводите число в каждый элемент массива:\n";
     for (i=0;i<M;i++)
       for (j=0;j<N;j++)
        {
         printf("a[%d][%d] = ",i,j); cin >> p[i][j];
        }
     break;
   case '2':
     cout << "\nВведите диапазон для случайных чисел (от 0 до r)  r = ";
     cin >> r;

     randomize();
     for (i=0;i<M;i++)
```

```
      for (j=0;j<N;j++) p[i][j]=random(r);
  }

cout << "\n\nМассив создан.";

getch();

return p;
}


//---------------------------------------------------------------------
// Выделение памяти под одномерный массив и заполнение его
//---------------------------------------------------------------------
int* CreateMass (unsigned int &sz)
{
int *p;
int i,r;
char punkt;

cout << "\n\nЗадайте размер массива ";
cin >> sz;
// Динамическое выделение памяти под массив
p=new int[sz];

if (!p)
  {
   cout << "\n\nОшибка! Невозможно создать массив! \nПопробуйте задать меньший
размер.";
   getch();
   return NULL;
  }

cout << "\n\nВвод значений в массив:";
cout << "\n1. Вручную";
cout << "\n2. Автоматически (случайные значения)\n";
punkt=getch();

switch (punkt)
  {
   case '1':
     for (i=0;i<sz;i++)
      {
       cout << "Ввод " << i <<"-го элемента: ";
       cin >> p[i];
      }
     break;
   case '2':
     cout << "\nВведите максимум для случайных чисел: ";
     cin >> r;
     randomize(); // Настройка генератора случайных чисел
     for (i=0;i<sz;i++)  { p[i]=random(r); }
  }

cout << "\n\nМассив создан.";

getch();
```

```
return p;
}

//---------------------------------------------------------------------
// Распечатка одномерного массива
//---------------------------------------------------------------------
void PrintMass (unsigned int sz, int* p)
{

int i;

if (!p)
  {
    cout << "\n\nОшибка! Массив еще не создан.";
    getch();
    return;
  }

cout << "\n\nМассив a[" << sz << "] содержит следующие значения:\n";
for (i=0;i<sz;i++) cout << p[i]<< "  ";
getch();

return;
}


//---------------------------------------------------------------------
// Сортировка части массива, определенной пользователем
//---------------------------------------------------------------------
void SortMass (unsigned int sz, int* p)
{
unsigned int k1,k2;
int buf;
int i,j;

if (!p)
  {
    cout << "\n\nОшибка! Массив еще не создан.";
    getch();
    return;
  }

// Ввод пользователем нач. и кон. индексов для сортировки
cout << "\n\n Сортировка начиная с индекса (число от 0 до "<< sz-1<< ") = ";
cin >> k1;
cout << " и заканчивая индексом (число от " << k1+1<<" до "<< sz-1<< ") = ";
cin >> k2;

// Вводимые значения должны быть от 0 до sz-1, первое значение должно быть меньше
второго!
if ((k1>=sz)||(k2>=sz)||(k1>=k2))
  {
   cout << "\n\nПараметры заданы неверно! Массив не отсортирован!";
   getch();
   return;
  }
```

```
// Сортировка по возрастанию части массива
for (i=k1;i<k2;i++)
 {
  for (j=i+1;j<=k2;j++)
   {
   if (p[i]>p[j])
     {
       buf=p[i];
       p[i]=p[j];
       p[j]=buf;
     }
   }
 }

cout << "\nЗаданная часть массива отсортирована.";

getch();

return;
}

//---------------------------------------------------------------------
// Нахождение максимального элемента массива
//---------------------------------------------------------------------
void MaxMass (unsigned int sz, int* p)
{

int max=0;
int i;

if (!p)
  {
    cout << "\n\nОшибка! Массив еще не создан.";
    getch();
    return;
  }


for (i=0;i<sz;i++)
  {
    if (p[max]<p[i]) {max=i;}
  }
cout << "\n\n Максимальное значение a["<< max<<"] = "<<p[max];

// Подсчет значений, равных максимальному
int lmax=0;

for (i=0;i<sz;i++)
  { if (p[i]==p[max]) {lmax++;} }

cout << "\n значений, равных максимальному в массиве " << lmax;

getch();
return;
}


//---------------------------------------------------------------------
```

```
// Нахождение суммы элементов после первого нуля
//-----------------------------------------------------------------
void SumAfterZerro (unsigned int sz, int* p)
{

int i;
int zer=-1; // индекс первого элемента равного нулю (если равен -1, то нуля в
массиве нет)
int sum=0; // сумма элементов после нуля

if (!p)
  {
    cout << "\n\nОшибка! Массив еще не создан.";
    getch();
    return;
  }

// Находим первый ноль и затем считаем сумму
for (i=0;i<sz;i++)
 {
  // Если индекс элемента равного нулю найден, то подсчет суммы
  if (zer!=-1) sum+=p[i];
  // ..иначе проверяем, является ли данный (i-тый) элемент нулевым
  else if (p[i]==0) {zer=i;}
 }

if (zer==-1) cout << "\n\n В массиве нет нулевых значений! Сумма не может быть
найдена!";
else  cout << "\n\n Сумма после нулевого элемента a["<<zer<<"] равна " << sum;

getch();
return;
}


//-----------------------------------------------------------------
// Подсчет количества строк, не содержащих ни одного нулевого элемента
//-----------------------------------------------------------------
void NonZerroRows (unsigned int M, unsigned int N, int** p)
{

unsigned int l,chk;
int i,j;

if (!p)
  {
    cout << "\n\nОшибка! Массив еще не создан.";
    getch();
    return;
  }


l=0;
for (i=0;i<M;i++)
  {
    chk=1; // Сначала предположим что i-я строка не содержит нулевых элементов
    for (j=0;j<N;j++)
      // Если найден нулевой элемент, то досрочно выйти из цикла
```

```
      if (p[i][j]==0)
        {
          chk=0;
          break;
        }
    // Если в ходе проверки нулевых элементов не найдено в строке, то l++
    if (chk) l++;
  }


cout << "\n В массиве найдено "<< l <<" строк без нулевых элементов";
getch();

return;

}


//-------------------------------------------------------------------
// Перестановка элементов столбцов, указанных пользователем
//-------------------------------------------------------------------
void SwitchColumn (unsigned int M, unsigned int N, int** p)
{

int i,j;
unsigned int k1,k2;
int buf;

if (!p)
  {
    cout << "\n\nОшибка! Массив еще не создан.";
    getch();
    return;
  }


printf ("\n Укажите номера столбцов k1 и k2 (от %d до %d), \n элементы которых
нужно поменять местами:\n",1,N);
cout << "k1 = "; cin >> k1;
cout << "k2 = "; cin >> k2;

// В цикле учитываем, что пользователем указаны номера (от 1 до N), а не индексы
(от 0 до N-1) столбцов

for (i=0;i<M;i++)
  {
   buf=p[i][k1-1];
   p[i][k1-1]=p[i][k2-1];
   p[i][k2-1]=buf;
  }

cout << "\n\n Все сделано. Элементы переставлены.";
getch();
return;

}
```

4. Результаты тестового прогона программы

Задача 1

1. Печать меню задачи:

```
ЗАДАЧА 1.
МЕНЮ
```

```
1. Создать массив
2. Распечатать массив
3. Подсчет максимальных значений
4. Сортировка части массива
5. Расчет суммы элементов после первого нуля
6. Вернуться в главное меню
```

2. Создание и распечатка массива (выбор пунктов меню 1 и 2):

Создание массива:

```
Задайте размер массива <ввод 20>



Ввод значений в массив:
1. Вручную
2. Автоматически (случайные значения)
<ввод 2>

Введите максимум для случайных чисел: <ввод 10>



Массив создан.
```

Распечатка массива:

```
Массив a[20] содержит следующие значения:
```

```
6  7  9  0  6  3  8  8  0  2  9  9  1  7  6  4  9  7  1  4
```

3. Подсчет максимальных значений (пункт меню 3):

```
Максимальное значение a[2] = 9
значений, равных максимальному в массиве 4
```

4. Сортировка по возрастанию элементов массива, начиная и заканчивая индексами, заданными пользователем (пункт меню 4):

Массив до сортировки (пункт меню 2):

```
Массив a[20] содержит следующие значения:
6  7  9  0  6  3  8  8  0  2  9  9  1  7  6  4  9  7  1  4
```

Сортировка (пункт меню 4):

```
Сортировка начиная с индекса (число от 0 до 19) = <ввод 4>
 и заканчивая индексом (число от 5 до 19) = <ввод 16>

Заданная часть массива отсортирована.
```

Массив после сортировки (пункт меню 2):

```
Массив a[20] содержит следующие значения:
6  7  9  0  0  1  2  3  4  6  6  7  8  8  9  9  9  7  1  4
```

5. Нахождение суммы элементов, расположенных после первого нуля (пункт меню 5):

```
Сумма после нулевого элемента a[3] равна 84
```

Вывод - все подпрограммы задачи 1 работают правильно.

Задача 2

1. Печать меню задачи:

```
   ЗАДАЧА 2.
   МЕНЮ
```

```
1. Создать двумерный массив
2. Распечатать массив
3. Подсчет строк без нулевых элементов
4. Поменять местами элементы столбцов
5. Вернуться в главное меню
```

2. Создание и распечатка массива (выбор пунктов меню 1 и 2):

Создание массива:

```
Задайте размер для двумерного массива
Строк в массиве:  M = <ввод 7>
Столбцов в массиве:  N = <ввод 9>


Ввод значений в массив:
1. Вручную
2. Автоматически (случайные значения)
<ввод 2>

Введите диапазон для случайных чисел (от 0 до r)  r = <ввод 25>


Массив создан.
```

Распечатка массива:

```
Массив a[7][9] имеет вид:
```

```
 12  13  22   3   9   7  19  20  23
 22   2  10   8   8   2  24   3  19
  2  22  10  10   1   6   6  20  15
 16  15   1   8   4  17  10   4  11
  6  24  20  23  23   5  15  11  18
 12   9  22   5  18  23  24   0   6
 12  17  17   1   5  11  11   4   0
```

3. Подсчет количества строк, не содержащих ни одного нулевого значения (пункт меню 3):

```
В массиве найдено 5 строк без нулевых элементов
```

4. Перестановка элементов двух столбцов, номера которых указаны пользователем (пункт меню 4):

```
Укажите номера столбцов k1 и k2 (от 1 до 9),
 элементы которых нужно поменять местами:
k1 = <ввод 3>
k2 = <ввод 7>


 Все сделано. Элементы переставлены.
```

Распечатка массива (пункт меню 2):

```
Массив a[7][9] имеет вид:
  12  13  19   3   9   7  22  20  23
  22   2  24   8   8   2  10   3  19
   2  22   6  10   1   6  10  20  15
  16  15  10   8   4  17   1   4  11
   6  24  15  23  23   5  20  11  18
  12   9  24   5  18  23  22   0   6
  12  17  11   1   5  11  17   4   0
```

Вывод - все подпрограммы задачи 2 работают правильно.

ЗАКЛЮЧЕНИЕ.

Программа была разработана в рамках семестрового задания, и обладает рядом достоинств и недостатков. Представляемый программный продукт отвечает все требованиям, предъявленным заказчиком (см. пункт «Постановка задачи»).

Разработка программного средства проводилась в полном соответствии с современными тенденциями в технологии программирования, а именно, с использованием принципов модульного и структурного проектирования программных средств. Для решения каждой из подзадач были разработаны отдельные функции, которые могут быть настроены на различные массивы с помощью изменения значений входных параметров. Данный подход существенно сократил программный код и, соответственно, память компьютера, необходимую для размещения и успешной работы программного средства, а так же увеличил быстродействие и эффективность работы программы.

Все задачи варианта объединены одним общим интерфейсом. Выполнение каждой задачи осуществляется как последовательный вызов всех необходимых подпрограмм, для этого был составлен уникальный алгоритм. Предусмотрена возможность повтора выполнения каждой задачи через пункты главного меню. Результаты печатаются с максимально возможными комментариями. Размерность массивов задается пользователем программного средства.

В случае необходимости программа сообщает об ошибках, возникающих по вине пользователя или при сбоях в самой программе. Например, при реализации решения подзадачи сортировки для определенной части массива, подпрограмма выполняет дополнительную проверку данных, вводимых пользователем, на корректность, что, соответственно, повышает эффективность всего программного продукта в целом.

Программа была разработана в процессе изучения разработчиком курса «Программирование и основы алгоритмизации», и поэтому обладает также и рядом недостатков. Были использованы не всегда оптимальные по времени исполнения алгоритмы для решения некоторых подзадач. Некоторые участки кода могут быть, по мнению автора, реорганизованы в более читабельный вид или реализованы более оптимально. Но, не смотря на эти недочеты, данная программная разработка, по твердому убеждению автора, полностью удовлетворяет заданным требованиям, и даже в некоторых моментах выходит за рамки задания, что является ее неоспоримым достоинством.

6. Использованная литература

1. Б.И.Березин, С.Б.Березин "Начальный курс С и С++" -М.: ДИАЛОГ-МИФИ, 1996 г.
2. В.А.Скляров "Язык С++ и объектно-ориентированное программирование" - Мн.: Выш. шк., 1997 г.
3. С.Поттс, Т.С.Монк "BORLAND С++ в примерах" - Мн.: ООО "Попурри", 1996 г.