Работа со строками. Понятия функций

Программа подсчета символов в строке

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6
7
8   char ourStr[128] = ""; // для сохранения строки
9
10  cout << "Введите текст латиницей (не больше 128 символов):\n";
11  cin.getline(ourStr, 128);
12
13  int amountOfSymbol = 0; // счетчик символов
14  while (ourStr[amountOfSymbol] != '\0')
15  {
16  amountOfSymbol++;
17  }
18
19  cout << "Строка \"" << ourStr << "\" состоит из "
20  << amountOfSymbol << " символов!\n\n";
21
22  return 0;
23 }
```

Поменяйте значение переменной *‘ourStr’ на задание по варианту.*

Повторение – функции работы со строками

К стандартным функциям библиотеки cstring относятся:

strlen() – подсчитывает длину строки (количество символов без учета \0);

strcat() – объединяет строки;

strcpy() – копирует символы одной строки в другую;

strcmp() – сравнивает между собой две строки

Повторение – функции работы со строками

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

int main()
{

char  line1[80] = “Я живу в стране Россия”;
char line2[80]=“А именно в городе Ульяновске”;
char line3[60]=“На улице …”;
char line4[60]=“В доме номер …”;
```

Повторение – функции работы со строками

```
cout<<“Строки line1=”<<line1;
cout<<“Строки line2=”<<line2;
cout<<“Строки line3=”<<line3;
cout<<“Строки line4=”<<line4;
//функция добавления одной строчки к другой
strcat (line1, line2);
// то что получилось после объединения
cout << “Объединим строчку line1 и line2” << line1;
//функция копирование одной строки и её вставку на место другой строки.
strcpy(line1,line2);
cout<<“Строка line1 после копирования в нее line2<<line1;
```

Повторение – функции работы со строками

```
cout<<“Строки line1 после изменения=”<<line1;
cout<<“Строки line2 после изменения=”<<line2;
cout<<“Строки line3 после изменения=”<<line3;
cout<<“Строки line4 после изменения=”<<line4;
return 0;
}
```

Разбейте проблему на части

В реальной жизни нам часто приходится выполнять очень сложные задачи. Понять, как их решить, также бывает очень трудно. В таких случаях можно использовать **метод деления на части** (или ***метод «от большого к малому»***). То есть, вместо того, чтобы решать одну большую сложную задачу, мы разбиваем её на несколько подзадач, каждую из которых проще решить. Если эти подзадачи все еще слишком сложные, то их также нужно еще раз разбить. И так до тех пор, пока мы не доберемся до точки, где каждая отдельно взятая задача — легко решаема.

Рассмотрим это на примере:

Допустим, что нам нужно написать доклад о картошке. Наша иерархия задач в настоящее время выглядит следующим образом:

✔ Написать доклад о картошке

Это довольно большая задача, давайте разделим её на подзадачи:

- Написать доклад о картошке
- Поиск информации о картошке
- Создание плана
- Заполнение каждого пункта плана подробной информацией
- Заключение

Это уже проще, так как теперь мы имеем список подзадач, на которых можем сосредоточиться в индивидуальном порядке.

В данном случае «Поиск информации о картошке» звучит немного расплывчато, нужно дополнительно разбить и этот пункт:

Написать доклад о картошке

✔ Поиск информации о картошке
✔ Сходить в библиотеку за книжками о картошке
✔ Поискать информацию в Интернете
✔ Делать заметки в соответствующих разделах из справочного материала

Создание плана

✔ Информация о выращивании
✔ Информация об обработке
✔ Информация об удобрениях

Заполнение каждого пункта плана подробной информацией

Заключение

Выполняя каждый подпункт этого задания, мы решим одну большую задачу.

Определите последовательность событий

Первый шаг заключается в определении последовательности событий.

Если бы мы создавали калькулятор, то последовательность заданий выглядела бы следующим образом:

- ✔ Получить первое значение от пользователя
- ✔ Получить математическую операцию от пользователя
- ✔ Получить второе значение от пользователя
- ✔ Вычислить результат
- ✔ Вывести результат

Этот список определяет содержимое функции main()

Определите последовательность событий

Расписать последовательность действий для выполнения заданий по варианту:

1. Купить книгу по географии через интернет
2. Написать доклад по астрономии
3. Заказать телефон на сайте
4. Съездить в гости к другу в другой город
5. Отправить решение примеров по электронной почте учительнице
6. Подключиться к конференции zoom

Определите последовательность событий

Первый шаг заключается в определении последовательности событий.

Если бы мы создавали калькулятор, то последовательность заданий выглядела бы следующим образом:

- ✔ Получить первое значение от пользователя
- ✔ Получить математическую операцию от пользователя
- ✔ Получить второе значение от пользователя
- ✔ Вычислить результат
- ✔ Вывести результат

Этот список определяет содержимое функции main()

Рассмотрим готовую версию программы-калькулятора, её структуру и перемещение данных:

```cpp
#include <iostream>

int getUserInput()
{
    std::cout << "Please enter an integer: ";
    int value;
    std::cin >> value;
    return value;
}

int getMathematicalOperation()
{
    std::cout << "Please enter which operator you want (1 = +, 2 = -, 3 = *, 4 = /): ";

    int op;
    std::cin >> op;

    // А что, если пользователь введет некорректный символ?
    // Пока что мы это проигнорируем

    return op;
}

int calculateResult(int x, int op, int y)
{
    // Обратите внимание, оператор == используется для сравнения двух значений

    if (op == 1) // если пользователь выбрал операцию сложения (№1)
        return x + y; // то выполняем эту строку
    if (op == 2) // если пользователь выбрал операцию вычитания (№2)
```

```
31          return x - y; // то выполняем эту строку
32      if (op == 3) // если пользователь выбрал операцию умножения (№3)
33          return x * y; // то выполняем эту строку
34      if (op == 4) // если пользователь выбрал операцию деления (№4)
35          return x / y; // то выполняем эту строку
36
37      return -1; // вариант, если пользователь ввел некорректный символ
38  }
39
40  void printResult(int result)
41  {
42      std::cout << "Your result is: " << result << std::endl;
43  }
44
45  int main()
46  {
47      // Получаем первое значение от пользователя
48      int input1 = getUserInput();
49
50      // Получаем математическую операцию от пользователя
51      int op = getMathematicalOperation();
52
53      // Получаем второе значение от пользователя
54      int input2 = getUserInput();
55
56      // Вычисляем результат и сохраняем его во временной переменной
57      int result = calculateResult(input1, op, input2 );
58
59      // Выводим результат
60      printResult(result);
61  }
```

Функции в C++

Функция — это последовательность стейтментов для выполнения определенного задания. Часто ваши программы будут прерывать выполнение одних функций ради выполнения других. Вы делаете аналогичные вещи в реальной жизни постоянно. Например, вы читаете книгу и вспомнили, что должны были сделать телефонный звонок. Вы оставляете закладку в своей книге, берете телефон и набираете номер. После того, как вы уже поговорили, вы возвращаетесь к чтению: к той странице, на которой остановились.

Программы на языке C++ работают похожим образом. Иногда, когда программа выполняет код, она может столкнуться с вызовом функции. **Вызов функции** — это выражение, которое указывает процессору прервать выполнение текущей функции и приступить к выполнению другой функции. Процессор «оставляет закладку» в текущей точке выполнения, а затем выполняет вызываемую функцию. Когда выполнение вызываемой функции завершено, процессор возвращается к *закладке* и возобновляет выполнение прерванной функции.

Возвращаемые значения

Когда функция main() завершает свое выполнение, она возвращает целочисленное значение обратно в операционную систему, используя **оператор return**.

Функции, которые мы пишем, также могут возвращать значения. Для этого нужно указать **тип возвращаемого значения** (или ***«тип возврата»***). Он указывается при объявлении функции, перед её именем. Обратите внимание, тип возврата не указывает, какое именно значение будет возвращаться. Он указывает только тип этого значения.

Тип возврата void

Функции могут и не возвращать значения. Чтобы сообщить компилятору, что функция не возвращает значение, нужно использовать тип возврата void.

Эта функция имеет тип возврата void, который означает, что функция не возвращает значения. Поскольку значение не возвращается, то и оператор return не требуется.