МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО АЛТАЙСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Физико-технический факультет

Кафедра информатики и вычислительной техники (ИВТ)

Лабораторная работа №08

**Структурный тип данных**

Выполнил студент 585 гр.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.М. Губченко

Проверил: к.т.н,, доцент каф. ВТиЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Ю.Г. Скурыдин

Лабораторная работа защищена

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019 г.

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Барнаул 2019

1. **Формулировка задачи**

Сформировать массив, содержащий сведения о количестве изделий категорий A, B, C, собранных рабочим за месяц. Структурный тип содержит поля: фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц. Считая заданным значения расценок SA, SB, SC за выполненную работу по сборке единицы изделия категорий A, B, C, вывести на экран общее количество изделий категорий A, B,C, собранных рабочим цеха

1. **Постановка задачи**

Инициализируется структура данных, содержащая поля: фамилия сборщика, наименование цеха, количество изделий по категориям, собранных рабочим за месяц, так же инициализируются три вещественных целых числа, определяющих цену одной детали категории A, B, C. На вход поступает символьная строка определяющая фамилию сборщика, символьная строка определяющая имя цеха, три вещественных числа определяющих количество изделий каждой из категорий. После этого программа выводит на экран 3 целых положительных числа, определяющее количество деталей каждой из категорий и целое положительное число определяющее их общее количество.

1. **Математическая модель**

Для построения математической модели рассмотрим пример.

Вводим фамилию «Ivanov», имя цеха «Soc», количество изделий категорий А = 15, В = 32, С = 60.

Вводим фамилию «Smirnov», имя цеха «AWDS», количество изделий категорий А = 2, В = 14, С = 0.

Вводим фамилию «Helstrem», имя цеха «Stroy», количество изделий категорий А = 50, В = 8, С = 3.

Программа считает общее количество деталий, получается 107. Так же программа считает количество денег выплаченных за детали каждой из категорий, и общую выплату. На выход поступает

Surname:Ivanov

Name c:Soc

All cat. A: 15  oplata: 150

All cat. B: 32  oplata: 608

All cat. C: 60  oplata: 2580

All cat.: 107  oplata: 3338

Surname:Smirnov

Name c:AWDS

All cat. A: 2  oplata: 20

All cat. B: 14  oplata: 266

All cat. C: 0  oplata: 0

All cat.: 16  oplata: 286

Surname:Helstrem

Name c:Stroy

All cat. A: 50  oplata: 500

All cat. B: 8  oplata: 152

All cat. C: 3  oplata: 129

All cat.: 61  oplata: 781

1. **Описание алгоритма**

**Начало алгоритма**

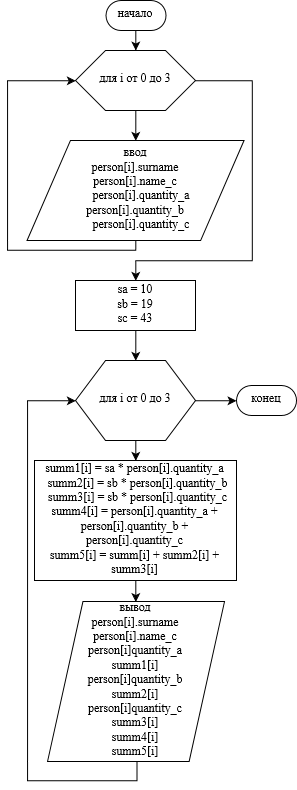
* 1. Присвоение переменной i значения 0
  2. Проверка условия, если i>=3, переход к пункту 4.9
  3. i++
  4. Ввод person[i].surname
  5. Ввод person[i].name\_c
  6. Ввод person[i].quantity\_a
  7. Ввод person[i].quantity\_b
  8. Ввод person[i].quantity\_c и переход к пункту 4.2
  9. Присвоение переменной sa значения 10
  10. Присвоение переменной sb значения 19
  11. Присвоение переменной sc значения 43
  12. Присвоение переменной i значения 0
  13. Проверка условия, если i>=3 конец выполнения программы
  14. i++
  15. Присвоение переменной summ1[i] значения sa \* person[i].quantity\_a
  16. Присвоение переменной summ2[i] значения sb \* person[i].quantity\_b
  17. Присвоение переменной summ3[i] значения sc \* person[i].quantity\_c
  18. Присвоение переменной summ4[i] значения person[i].quantity\_a + person[i].quantity\_b + person[i].quantity\_c
  19. Присвоение переменной summ5[i] значения summ1[i] + summ2[i] + summ3[i]
  20. Вывод person[i].surname
  21. Вывод person[i].name\_c
  22. Вывод person[i].quantity\_a
  23. Вывод summ1[i]
  24. Вывод person[i].quantity\_b
  25. Вывод summ2[i]
  26. Вывод person[i].quantity\_c
  27. Вывод summ3[i]
  28. Вывод summ4[i]
  29. Вывод summ5[i] и переход к пункту 4.13

**Конец алгоритма**

1. **Проект программы**

В данном случае программа должна состоять из 27 частей: цикл с параметром(два блока арифметических действий, блок сравнения, 5 блоков ввода), три блока арифметических действий, цикл с параметром( 7 блоков арифметических действий, 6 блоков вывода, блок проверки условия)

1. **Опорный граф (блок-схема) алгоритма**



1. **Текст программы:**

**на языке С++**

#include <iostream>

#include <string>

using namespace std;

struct info

{

char surname [10];

char name\_c [15];

int quantity\_a;

int quantity\_b;

int quantity\_c;

};

int main()

{

info person [3];

for (int i = 0; i < 3; i++)

{

cout << "Surname :";

cin >> person[i].surname;

cout << "Name c.:";

cin >> person[i].name\_c;

cout << "quantity A:";

cin >> person[i].quantity\_a;

cout << "quantity B:";

cin >> person[i].quantity\_b;

cout << "quantity C:";

cin >> person[i].quantity\_c;

}

int sa, sb, sc;

sa = 10;

sb = 19;

sc = 43;

int summ1[3], summ2[3], summ3[3], summ4[3], summ5[3];

for (int i = 0; i < 3; i++)

{

summ1[i] = sa \* person[i].quantity\_a;

summ2[i] = sb \* person[i].quantity\_b;

summ3[i] = sb \* person[i].quantity\_c;

summ4[i] = person[i].quantity\_a + person[i].quantity\_b + person[i].quantity\_c;

summ5[i] = summ1[i] + summ2[i] + summ3[i];

cout << "Surname:" << person[i].surname << '\n';

cout << "Name c:" << person[i].name\_c << '\n';

cout << "All cat. A: " << person[i].quantity\_a << " oplata: " << summ1[i] <<'\n';

cout << "All cat. B: " << person[i].quantity\_b << " oplata: " << summ2[i] <<'\n';

cout << "All cat. C: " << person[i].quantity\_c << " oplata: " << summ3[i] <<'\n';

cout << "All cat.: " << summ4[i] << " oplata: " << summ5[i] <<'\n';

cout << " " << '\n';

}

return 0;

}

1. **Проверка работоспособности (тестирование) программы**

Для проверки работоспособности программы необходимо просто посмотреть на то, что выведет программа в конце работы.

1. **Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

Ввиду простоты программы она одинаково эффективно выполняется для обоих языков программирования.

1. **Формулировка задачи**

Дана строка слов, разделённых пробелами и запятыми. Подсчитайте количество слов в строке и сформируйте новую строку из самых длинных слов подстрок (заключённых между запятыми)

1. **Постановка задачи**

На вход программы поступает символьная строка слов, содержащая также пробелы и запятые. Программа посчитывает количество слов в строке и формирует новую строку из самых длинных слов подстрок, заключённых между запятыми. На выход поступает символьная строка и целое положительное число определяющее количество слов в исходной строке.

1. **Математическая модель**

Для построения математической модели рассмотрим пример. Инициализируется строка «Hi, how are you doing with the session». После этого программа проверяет каждый символ на то, является ли он пробелом или запятой, если не является, то она добавляет его в другую строку. Таким образом программа запоминает всё слово и его длину. Если встречается пробел, то полученная строка сохраняется, и длина данного слова считается максимальной. После этого программа считает длину нового слова. Если длина нового слова больше предыдущего, то оно становится максимальным. Если встречается запятая, то программа сравнивает длину найденного слова с максимумом и если это слово больше максимального, то оно выводится на экран, иначе выводится максимальное. После этого очищается буфер, который хранит саме длинное слово и количество букв в нём. Далее программа ищет самое длинное слово в новой подстроке используя алгоритм описанный выше. В данном случае программы выведет строку «Hi session» и число 8.

1. **Описание алгоритма**

**Начало алгоритма**

* 1. Присвоение переменной temp значения 0
  2. Присвоение переменной maxlen значения 0
  3. Присвоение переменной j значения 0
  4. Присвоение переменной word значения 0

// ввод строки Str

* 1. getline(cin, Str)
  2. Присвоение переменной i значения 0

//цикл по строке

* 1. Проверка условия, если i>=Str.size(), то переход к пункту 4.32
  2. i++

//если запятая

* 1. Проверка условия, если Str[i] != ‘,’ то переход к пункту 4.19
  2. Проверка условия, если temp<=maxlen, то переход к пункту 4.13
  3. Присвоение переменной maxlen значения temp
  4. Присвоение переменной MaxW ordзначения TempWord
  5. Вывод MaxWord
  6. MaxWord.clear()
  7. memset(TempWord, 0, sizeof(TempWord))
  8. Присвоение переменной temp значения 0
  9. Присвоение переменной j значения 0
  10. Присвоение переменной maxlen значения 0

//если это символ

* 1. Проверка условия, если Str[i]==’ ‘ и Str[i]==’,’ то переход к пункту 4.25
  2. Temp++
  3. Присвоение переменной TempWord[j] значения Str[i]
  4. j++
  5. Проверка условия, если Str[i+1]!=’ ‘ илиStr[i+1]!=’,’ то переход к пункту 4.25
  6. word++

//если пробел

* 1. Проверка условия, если Str[i]!=’ ‘, то переход к пункту 4.7
  2. Проверка условия, если temp<=maxlen, то переход к пункту 4.29
  3. Присвоение переменной maxlen значения temp
  4. Присвоение переменной MaxWord значения TempWord
  5. memset(TempWord, 0, sizeof(TempWord))
  6. Присвоение переменной temp значения 0
  7. Присвоение переменной j значения 0

//конец выполнения

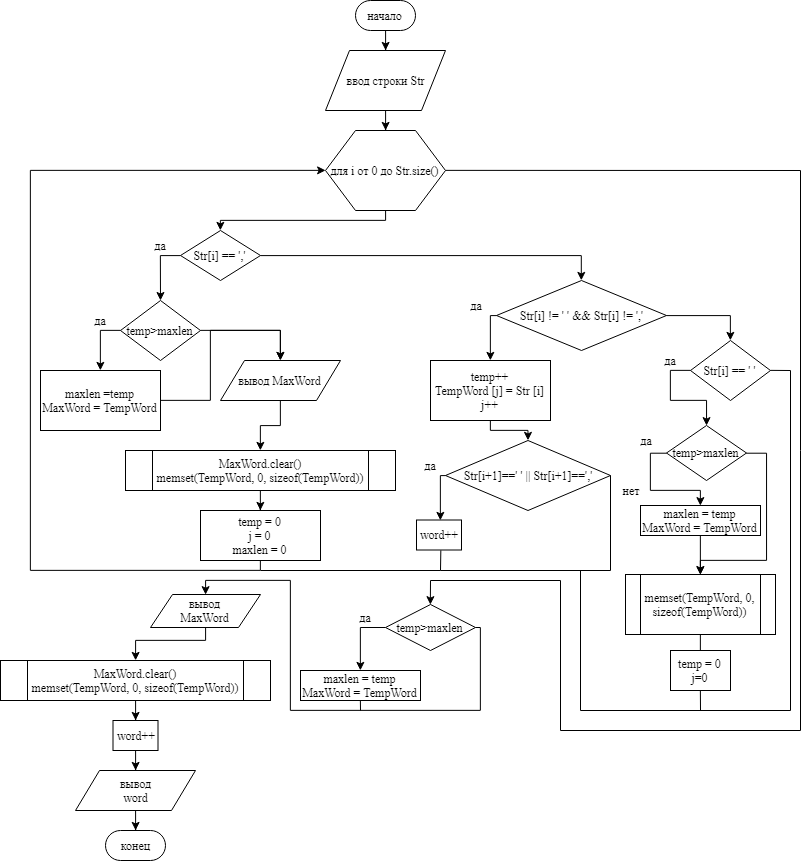
* 1. Проверка условия, если temp<= maxlen, то переход к пункту 4.35
  2. Присвоение переменной maxlen значения temp
  3. Присвоение переменной MaxWord значения TempWord
  4. Вывод MaxWord
  5. MaxWord.clear()
  6. memset(TempWord, 0, sizeof(TempWord))
  7. word++
  8. вывод word

**Конец алгоритма**

1. **Проект программы с подсчётом слов в строке, и выводом самых длинных слов подстрок**

В данном случае программа должна состоять из 53 частей: 6 блоков арифметических действий, два блока ввода цикл с параметром(7 блоков проверки условия, 15 блоков арифметических действий, один блок вывода, 3 стандартные функции), блок проверки условия, 3 блока арифметических действий, два блока вывода, две стандартные функции.

1. **Опорный граф (блок-схема)**



1. **Текстпрограммы:**

**на языке С++**

#include <iostream>

#include <cstring>

**usingnamespace**std;

**int**main()

{

string Str;//исходная строка

// string TempWord; //текущее слово

**char** TempWord [20];

string MaxWord;//максим. слово

MaxWord [0]='\0';

**char** ClearStr;

ClearStr =' ';

**int** temp =0;

**int** maxlen =0;

**int** j=0;

**int** word =0;

cout<<"enter the string:"<<'\n';

getline(cin, Str);//вводстроки

**for**(size\_t i=0; i<Str.size(); i++)//циклповсейстроке

{

**if**(Str[i]==',')// если запятая, то

{

**if**(temp>maxlen)// если длина текущего слова больше максимальной длины, то

{

maxlen = temp;// обновление максимальной длины

MaxWord = TempWord;// обновление буфера максимальных слов

// cout << "MaxWord:" << MaxWord << '\n';

}

cout<< MaxWord <<"";// вывод текущего макс.слова

MaxWord.clear();//очистка буфера максим.слов

memset(TempWord,0,**sizeof**(TempWord));// очистка буфера текущего слова

//cout << "Temp clear" << '\n';

temp =0;

j=0;

maxlen =0;

}

**if**(Str[i]!=' '&& Str[i]!=',')// если это не пробел и не запятая, то

{

temp++;// кол-во букв в слове +1

TempWord [j]= Str [i];// запись символа в буфер слов

//cout << "TempWord: " << TempWord << '\n';

j++;

**if**(Str[i+1]==' '|| Str[i+1]==','){

word++;

}

}

**if**(Str[i]==' ')// если это пробел, то

{

**if**(temp>maxlen)// если длина текущего слова больше максимальной длины, то

{

maxlen = temp;// обновление максимальной длины

MaxWord = TempWord;// обновление буфера максимальных слов

// cout << "MaxWord:" << MaxWord << '\n';

}

memset(TempWord,0,**sizeof**(TempWord));

//cout << "Temp clear" << '\n';

temp =0;

j=0;

}

}

**if**(temp>maxlen)// если длина текущего слова больше максимальной длины, то

{

maxlen = temp;// обновление максимальной длины

MaxWord = TempWord;// обновление буфера максимальных слов

//cout << "MaxWord:" << MaxWord << '\n';

}

cout<< MaxWord <<"";// вывод текущего макс.слова

MaxWord.clear();//очистка буфера максим.слов

memset(TempWord,0,**sizeof**(TempWord));// очистка буфера текущего слова

//cout << "Temp clear" << '\n';

word++;

cout<<""<<'\n';

cout<<"Word= "<< word <<'\n';

**return**0;

}

1. **Проверкаработоспособности (тестирование) программы**

Для проверки работоспособности программы необходимо посмотреть на то, что выведет программа.

1. **Сравнительный анализ и оценка эффективности работы программ на разных языках программирования**

Ввиду простоты программы она одинаково эффективно выполняется для обоих языков программирования.