Прототипы функций обработки нуль–терминированных строк:

**size\_t strlen(const char \*s);**// возвращает длину строки s без учёта завершающего //символа ‘\0’

**char\* strcpy(char \*s1, const char \*s2)**;// копирует s2 в s1, включая завершающий

//символ ‘\0’

**char\* strncpy(char \*s1, const char \*s2, size\_t n**);// копирует не более n символов

//из s2 в s1

**int strcmp(const char \*s1, const char \*s2)**;//сравнивает строки s1 и s2 и возвра–

//щает значение 0, -1, 1, если s1 соответственно равна, меньше или больше, чем s2

**int strncmp(const char \*s1, const char \*s2, size\_t n)**;//сравнивает не более n

//символов строки s1 со строкой s2 и возвращает одно из значений 0, -1 или 1

**char\* strcat(char \*s1, const char \*s2)**;// добавляет строку s2 к строке s1, причём

//первый символ строки s2 записывается поверх завершающего нуля строки s1

**char\* strncat(char \*s1, const char s2\*, size\_t n)**;// добавляет не более n символов

//строки s2 в строку s1

**char\* strtok(char \*s1, const char \*s2)**;// Последовательность вызовов этой

//функции разбивает строку s1 на лексемы, разделённые символами,

//содержащимися в строке s2. Первый вызов содержит в качестве первого

//параметра s1, а последующие вызовы для продолжения обработки той же

//строки содержат в качестве первого параметра 0. При каждом вызове

//возвращается указатель на текущее слово. Если при вызове функции слов

//больше нет, возвращается 0. Следует подчеркнуть, что функция strtok() изменяет

//первый параметр – строку s1.

//Рассмотрим пример использования функции **strtok():**

**char \*s1=new char [100];**

**strcpy(s1, "white\*^green^~~ black~^\*red ")**;//s1– строка, из которой надо выделить слова

**char \*s2="~\*^", \*w;** //s2 – строка с разделителями между словами

**w = strtok(s1, s2);**//выделяем из строки s1 первое слово и его адрес заносим в w

**while(w != NULL)** // пока в строке s1 есть слова, выделяем их и выводим на экран

**{**

**cout << w << endl;**

**w = strtok(NULL, s2);**

**}**

Прототипы функций для поиска в нуль–терминированных строках:

**char\* strchr(const char \*s, int c);//** определяет позицию первого вхождения

//символа c в строку s; если c найден, возвращается указатель на c в s; в

//противном случае возаращается указатель 0

**size\_t strcspn(const char \*s1, const char \*s2);//** определяет и возвращает длину //начальной части строки s1,

состоящей из символов, не содержащихся в s2

**size\_t strspn(const char \*s1, const char \*s2);//** определяет и возвращает длину //начальной части строки s1,

состоящей только из символов, содержащихся в s2

**char\* strpbrk(const char \*s1, const char \*s2**);// определяет позицию первого //вхожденияв в s1 любого из

символов строки s2

**char\* strstr(const char \*s1, const char \*s2**);// определяет позицию первого

//вхождения в строку s1 подстроки s2

Прототипы функций преобразования нуль–терминированных строк:

//Функции преобразования строк объявлены в библиотеке <stdlib.h>.

//Функции **atoi(), atol(), atof()** преобразуют символьную строку s в число в

//соответствии с типом возвращаемого значения. Символы пробела в начале

//строки пропускаются. Если указан знак, то он учитывается. Если встречается //недопустимый символ, то

преобразование прекращается и возвращаемым

//значением является число, полученное преобразованием символов с начала

//строки до первого недопустимого символа. Если недопустимым является первый //символ строки, то

возвращается число 0.

int atoi(const char \*s); // ascii to int

long atol(const char \*s); // ascii to long

double atof(const char \*s);// ascii to float

//Функции **strtol(), strtoul()** преобразуют символьную строку s в целое число,

//используя систему счисления с основанием basis. Преобразование прекращается,

//если всречается символ, который не может быть распознан как составная часть

//числа в заданной системе счисления. Указатель на этот символ записывается в //параметр endptr.

**long strtol(const char \*s, char \*\*endptr, int basis);**

**unsigned long strtoul(const char \*s, char \*\*endptr, int basis);**

//**Функция strtod**() преобразует символьную строку s в число типа double.

//Указатель endptr отмечает символ, на котором была прекращена обработка.

**double strtod(const char \*s, char \*\*endptr);**

//Функции **itoa(), ltoa(), ultoa()** преобразуют целое число value в нуль–

//терминированную символьную строку s, используя систему счисления с

//основание basis.

char\* itoa(int value, char \*s, int basis); //int to ascii

char\* ltoa(long value, char \*s, int basis); //long to ascii

char\* ultoa(unsigned long value, char \*s, int basis);//unsigned long to ascii

//Функция gcvt() преобразует value – число с плавающей точкой типа double в

//нуль–терминированную символьную строку s. Параметр digits задаёт количество

//значащих цифр числа value, которые будут помещены в строку s.

char\* gcvt(double value, int digits, char \*s);

Прототипы функций классификации символов:

//Функции классификации ASCII–символов объявлены в библиотеке <ctype.h>.

//int isalpha(int c); // буква: ‘a’..’z’ и ’A’..’Z’

//int isupper(int c); // буква в верхнем регистре: ‘A’..’Z’

//int islower(int c); // буква в нижнем регистре: ‘a’..’z’

//int isdigit(int c); // ‘0’..’9’

//int isxdigit(int c); // ‘0’..’9’ или буква

//int isspace(int c); // символы–разделители

//int iscntrl(int c); // управляющие символы (ASCII 0..31 и 127)

//int ispunct(int c); // пунктуация: ни один символ из вышеупомянутых

//int isalnum(int c); // isalpha() или isdigit()

//int isprint(int c); // то, что можно напечатать

//int isgraph(int c); // isalpha() или isdigit() или ispunct()

//int toupper(int c); // эквивалент в верхнем регистре

//int tolower(int c); // эквивалент в нижнем регистре

**СТРОКИ ТИПА string**

Средства стандартной библиотеки для работы со строками типа string становятся доступными

при подключении заголовочного файла <string>. Рассмотрим основные аспекты работы со строками

типа string.

Строку string можно инициализировать нуль–строкой, другой строкой string, подстрокой строки string

или последовательностью символов. Однако её нельзя инициализировать символом или числом. Например,

string s0; // пустая строка

string s1 = "Hello, wolrd!";// инициализация строковой константой

char s[] ="Hello, wolrd!"; // нуль–строка

string s11(s); // так можно преобразовывть нуль–строки в string

string s12 = s; // так тоже можно преобразовывть нуль–строки в string

string s2 = s1; // копия s1, можно записать и так: string s2(s1);

string s3(s2, 0, 5); // 5 символов из s2, начиная с первого, то есть "Hello"

string s4(7, 'q'); // 7 символов 'q', то есть "qqqqqqq"

Нельзя объявить строку длиной n. В случае такой необходимости можно объявить строку, в которую

поместить n копий указанного символа (строка s4).

Длина строки определяется функцией **length(),** например, s4.length() вернёт значение 7. Отметим, что

вычисление длины строки не основывается на понятии «завершающего нуля», обрывающего нуль–строку.

Реализация string хранит длину строки, не полагаясь на завершающий символ '\0'. Символы в строке нумеруются, начиная с 0, так что строка – это последовательность символов с номерами от 0 до length()–1.

Над строками можно выполнять простейшие операции:

присваивание: =

конкатенация: + , += ,

сравнение: = = , < , > , >= , <=,

индексирование: [] ,

ввод–вывод: >>, << .

Эти операции позволяют использовать объекты string в нормальных выражениях. В выражениях можно смешивать строки string и нуль–строки. Следует заметить, что при присваиваниях и конкатенациях результирующая строка string будет увеличиваться по мере надобности в соответствии с размером новой строки. Строке string также можно присвоить один символ или добавить один символ. 2

Далее приводятся функции – члены класса string. Используются следующие обозначения: pos означает позицию, а n – количество символов в строке – объекте класса string, для которой вызывается функция; pos1 означает позицию, а n1 – количество символов в строке, которая передаётся функции как параметр; npos означает «все элементы строки».

*Присваивание строк и частей строк*

string& assign(const string& s); //например, s4.assign(s1);

string& assign(const string& s, size\_t pos1, size\_t n1); //s4.assign(s1, 7, 5);

string& assign(const char\* p);//например, s4.assign(s);

string& assign(const char\* p, size\_t n1);//например, s4.assign(s, 5);

string& assign(size\_t n1, char c);//например, s4.assign(5, 'q');

*Добавление строк и частей строк*

string& append(const string& s);

string& append(const string& s, size\_t pos1, size\_t n1);

string& append(const char\* p);

string& append(const char\* p, size\_t n1);

string& append(size\_t n1, char c);

*Вставка строк и частей строк*

string& insert(size\_t pos, const string& s);

string& insert(size\_t pos, const string& s, size\_t pos1, size\_t n1);

string& insert(size\_t pos, const char\* p);

string& insert(size\_t pos, const char\* p, size\_t n1);

string& insert(size\_t pos, size\_t n1, char c);

*Замена в строках*

string& replace(size\_t pos, size\_t n, const string& s);

string& replace(size\_t pos, size\_t n, const string& s, size\_t pos1, size\_t n1);

string& replace(size\_t pos, size\_t n, const char\* p);

string& replace(size\_t pos, size\_t n, const char\* p, size\_t n1);

string& replace(size\_t pos, size\_t n, size\_t n1, char c);

*Удаление подстроки*

string& erase(size\_t pos = 0, size\_t n = npos); 3

Поиск подстрок

//Функции возвращают номер позции в строке, если поиск завершился успешно, и //–1 в противном случае.

*//Поиск первого вхождения подстроки в строку*

size\_t find(const string& s, size\_t pos = 0) const;

size\_t find(const char\* p, size\_t pos = 0) const;

size\_t find(const char\* p, size\_t pos, size\_t n1) const;

size\_t find(const char c, size\_t pos = 0) const;

*//Поиск последнего вхождения подстроки в строку*

size\_t rfind(const string& s, size\_t pos = npos) const;

size\_t rfind(const char\* p, size\_t pos = npos) const;

size\_t rfind(const char\* p, size\_t pos, size\_t n1) const;

size\_t rfind(const char c, size\_t pos = npos) const;

*//Поиск первого вхождения любого символа подстроки в строку*

size\_t find\_first\_of(const string& s, size\_t pos = 0) const;

size\_t find\_first\_of(const char\* p, size\_t pos = 0) const;

size\_t find\_first\_of(const char\* p, size\_t pos, size\_t n1) const;

size\_t find\_first\_of(const char c, size\_t pos = 0) const;

*//Поиск последнего вхождения любого символа подстроки в строку*

size\_t find\_last\_of(const string& s, size\_t pos = npos) const;

size\_t find\_last\_of(const char\* p, size\_t pos = npos) const;

size\_t find\_last\_of(const char\* p, size\_t pos, size\_t n1) const;

size\_t find\_last\_of(const char c, size\_t pos = npos) const;

*//Поиск в строке первого символа, которого нет подстроке*

size\_t find\_first\_not\_of(const string& s, size\_t pos = 0) const;

size\_t find\_first\_not\_of(const char\* p, size\_t pos = 0) const;

size\_t find\_first\_not\_of(const char\* p, size\_t pos, size\_t n1) const;

size\_t find\_first\_not\_of(const char c, size\_t pos = 0) const;

*//Поиск в строке последнего символа, которого нет подстроке*

size\_t find\_last\_not\_of(const string& s, size\_t pos = npos) const;

size\_t find\_last\_not\_of(const char\* p, size\_t pos = npos) const;

size\_t find\_last\_not\_of(const char\* p, size\_t pos, size\_t n1) const;

size\_t find\_last\_not\_of(const char c, size\_t pos = npos) const;

Сравнение строк

// Функции возвращают значения –1, 0, 1, если строка – объект, для которой //вызывается функция,

соответственно меньше, равна или больше строки, которая //передаётся функции как параметр.

int compare(const string& s)const;

int compare(size\_t pos, size\_t n, const string& s)const;

int compare(size\_t pos, size\_t n, const string& s, size\_t pos1, size\_t n1,)const;

int compare(const char\* p)const;

int compare(size\_t pos, size\_t n, const char\* p, size\_t n1 = npos)const;

Функция **getline()**

istream& getline(istream&, string& s);

istream& getline(istream&, string& s, char term);

Преобразование в нуль–строку

const char\* c\_str() const;

Перевод из типа stringв тип char\* достаточно сложен. Класс stringимеет в своем составе метод c\_str,

который возвращает указатель на нуль-терминированную строку, однако дополнительная память при

этом не выделяется, метод имеет прототипconstchar\*c\_str() const;т.е. он возвращает указатель на

константу. Для полноценной работы с такой строкой необходимо записать следующий код:strings1

;// получим указатель на строку константуconstchar\*sc1 = s1.c\_str();char\*sc2=newchar[strlen(sc1)+1];strcpy(sc2, sc1);

// и далее работаем со строкой sc2

Еще один вариант реализации этого механизма: char \*sc2= new char[s1.length()+1]; strcpy(sc2, s1.c\_str()));

С другой стороны, запись char \*sc2 = sc1; приведет к ошибке компиляции из-за неверного преобразования типов.

**Пример 1.** С клавиатуры вводится строка. Строка разделена на слова символами–разделителями. Разработать программу, которая будет переставлять в строке первое слово мак. длины и посл. словом мин. длины.

#include <iostream>

#include <string>

using namespace std;

int main()

{

string str, s2; //объявляем строку, в которую будем читать с клавиатуры

string d = " ,;:"; //строка с символами-разделителями

string s; //для слова

unsigned pos1 = 0, pos2, imax, imin;

cout << "Input your string" << endl;

getline(cin, str);

s2 = str; //копия строки str

string smax = "";

string smin = str;

str = str + d; //завершаем строку символами–разделителями

//Пока в исходной строке есть символы, отличные от символов–разделителей,

//выполняем цикл:

while ((pos1 = str.find\_first\_not\_of(d, pos1)) != string::npos)

{

//Выполним в строке str, начиная с позиции pos1, поиск первого разделителя.

pos2 = str.find\_first\_of(d, pos1);

//Копируем в строку s символы строки str, начиная с позиции pos1, в количестве

//pos2 – pos1.

s = str.substr(pos1, pos2 - pos1);//или так: s.assign(str, pos1, pos2– pos1);

if( s.length() > smax.length() )

{

smax = s; imax = pos1;

}

if (s.length() <= smin.length())

{

smin = s; imin = pos1;

}

pos1 = pos2;

}

//обмен словами

str = s2;

int lmax = smax.length();

int lmin = smin.length();

if (imax > imin)

{

str.replace(imax, lmax, smin);

str.replace(imin, lmin, smax);

}

else

if (imax < imin)

{

str.replace(imin, lmin, smax);

str.replace(imax, lmax, smin);

}

cout << str << endl;

return 0;

}

Пример 2. С клавиатуры вводится строка. Строка разделена на слова символами–разделителями.

Разработать программу, которая будет искать в заданной строке слова, представляющие собой

корректные записи числовых десятичных констант языка C++.

#include <iostream>

#include <cstdlib>

#include <string>

#include <cstring> //библиотека для работы с нуль–строками

//Тип string определён в пространстве имён std стандартной библиотеки C++.

//Объявляем все имена пространства std глобальными.

using namespace std;

void main()

{

string str; //объявляем строку, в которую будем читать с клавиатуры

string d = " ,;:"; //строка с символами-разделителями

char w[100]; //в нуль–строку w будем заносить выделенное слово, чтобы

//затем использовать функцию strtod() библиотеки <cstring> для преобразования

//слова в число

string s; //в строку s будем заносить слово, выделенное из строки str

unsigned pos1 = 0, pos2;

char \*p; //указатель на строку

cout << "Input your string" << endl;

getline(cin, str);

str = str + d; //завершаем строку символами–разделителями

//Пока в исходной строке есть символы, отличные от символов–разделителей,

//выполняем цикл:

while ( (pos1 = str.find\_first\_not\_of(d, pos1) ) != string::npos)

{

//Выполним в строке str, начиная с позиции pos1, поиск первого разделителя.

pos2 = str.find\_first\_of(d, pos1);

//Копируем в строку s символы строки str, начиная с позиции pos1, в количестве

//pos2 – pos1.

s = str.substr(pos1, pos2 - pos1);//или так: s.assign(str, pos1, pos2– pos1);

//Преобразуем строку s из типа string в нуль–строку, которую копируем в w.

strcpy(w, s.c\_str());

//Следующие две строки комментариев демонстрируют ещё одну возможность

//выделения слова из строки str и преоразования его к типу char\*.

//str.copy(w, pos2 – pos1, pos1); копируем символы строки str в w

//w[pos2– pos1] = '\0'; завершаем строку w символом '\0'

//Преобразуем строку w в число без сохранения результата преобразования.

strtod(w, &p);

//Функция strtod() вернёт в указателе p позицию первого непреобразованного

//символа в строке w. Если все символы строки w преобразованы в число, то

//p[0]=='\0', т.е. strlen(p) == 0. Если длина строки p равна 1, то символ p[0] может

//быть суффиксом в записи числовой десятичной константы. Далее проверяем это.

if ((strlen(p) == 0) || (strlen(p) == 1 && (p[0] == 'l' || p[0] == 'L' ||

p[0] == 'u' || p[0] == 'U' || p[0] == 'f' || p[0] == 'F')))

cout << w << endl;

pos1 = pos2;

}

}