1. Написати функцію, яка визначає, чи є рядок паліндромом (тобто рядком, який можна читати і зліва направо, і праворуч наліво.
2. Написати функцію, яка перетворює рядок на ціле число. Якщо рядок містить не цифрові символи – підрахунок зупиняється. Так рядок 123abc дасть число 123, а рядок a23f дасть число 0.
3. Напишіть програму «Телефонний довідник». Інформація зберігається у двомірному масиві рядків. Перша розмірність зберігає імена, друга – телефонні номери. Реалізувати меню з можливістю додавання, видалення, перегляду записів та сортування записів по прізвищу.
4. Користувач вводить з клавіатури арифметичний вираз. Необхідно обчислити його значення з урахуванням пріоритетів запроваджених математичних операцій та дужок. Якщо у виразі зустрічаються інші символи, видати повідомлення, що вираз не коректний.
5. \*вираз містить операції +, - та числа складаються з однієї цифри
6. \*\*вираз містить операції +, -,\*,/ та числа можуть складатися з кількох цифр
7. \*\*\* вираз містить операції +, -, \*, /, ^ (зведення в ступінь), дужки «(» «)» і числа можуть бути речовими.

Завдання 1

#include<string.h>

#include <iostream>

#include <stdio.h>

using namespace std;

#define size 100

bool Polindrom(char str[])

{

bool b = true;

for (int i = 0, j = strlen(str) - 1; i < strlen(str) / 2; i++, j--)

{

if (str[i] != str[j])

{

b = false;

break;

}

}

return b;

}

char\* Polindrom2(char\* str)

{

bool b = true;

for (int i = 0, j = strlen(str) - 1; i < strlen(str) / 2; i++, j--)

{

if (str[i] != str[j])

{

b = false;

break;

}

}

if (b)

return (char\*)"polindrom";

else

return (char\*)"nepolindrom";

}

int main()

{

setlocale(0, "");

char str[200];

cout << "Введiть рядок:\n";

cin.getline(str, 200);

cout << Polindrom(str) << endl;

char\* polindrom2 = Polindrom2(str);

cout << polindrom2 << endl;

return 0;

}

Завдання 2

#include<string.h>  
#include <iostream>  
#include <stdio.h>  
#include <stdlib.h>  
using namespace std;  
#define size 100

int Stroka\_v\_int(char str[])  
{  
int a = 0, k=0;  
char chisla[size];

if (str[0] >= '0' && str[0] <= '9')  
{  
for (int i = 0; i < strlen(str) - 1; i++)  
{  
if (str[0] >= '0' && str[0] <= '9')  
{  
chisla[k] = str[i];  
chisla[k + 1] = '\0';  
k++;  
}  
else  
break;

}  
a = atoi(chisla);  
}  
return a;  
}

int main()  
{  
setlocale(0, "");  
char str[size];  
cout << "Введiть рядок:\n";  
cin.getline(str, size);  
  
cout << Stroka\_v\_int(str);  
return 0;  
}

Завдання 3

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<string.h>

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

#define size 100

void Delete\_im9\_i\_tel(char str[][size], int index, int \*k)

{

char temp[size][size];

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

{

strcpy(temp[i], str[i]);

}

int k2 = index + 1;

for (int i = index; i < \*k; i++)

{

strcpy(temp[i], str[k2++]);

}

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

{

for (int j = 0; j < size; j++)

{

str[i][j] = temp[i][j];

}

}

(\*k)--;

}

int main()

{

setlocale(0, "");

int a, k = 2;

bool b;

char name[size][size] = { "Petro","Anton" }, tel[size][size] = { "+380989324004", "0662609970" };

cout << "Телефонний довiдник\n";

cout << "Меню:\n";

cout << "Додати iм'я та номер телефону - |1|, видалення по iменi - |2|, переглянути телефонний довiдник - |3|, сортування по прiзвищу - |4|, вихiд - |0|\n";

cin >> a;

for (; a != 0;)

{

switch (a)

{

case 1:

cout << "Введiть iм'я ->";

cin>>name[k];

cout << "Введiть номер телефону->";

cin>>tel[k];

k++;

break;

case 2:

b = false;

char del\_name[size];

cout << "Введiть iм'я, яке хочете видалити ->";

cin>>del\_name;

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

{

if (strcmp(del\_name, name[i]) == 0)

{

b = true;

Delete\_im9\_i\_tel(name, i, &k);

k++;

Delete\_im9\_i\_tel(tel, i, &k);

}

}

if (b == false)

cout << "Немає такого iменi\n";

break;

case 3:

int s;

cout << "Переглянути весь - |1|, переглянути по iменi - |2|\n";

cout << "->";

cin >> s;

switch (s)

{

case 1:

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

{

cout << name[i] << endl;

cout << tel[i] << endl;

cout << "-----------------------\n";

}

break;

case 2:

b = false;

char name\_look\_for[size];

cout << "Введiть iм'я, яке хочете знайти\n";

cout << "->";

cin >> name\_look\_for;

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

{

if (strcmp(name\_look\_for, name[i]) == 0)

{

cout << name[i] << endl;

cout << tel[i] << endl;

cout << "-----------------------\n";

b = true;

}

}

if (b == false)

cout << "Немає такого iменi\n";

break;

}

break;

case 4:

do {

b = false;

for (int i = 0; i < k-1; i++) {

if (strcmp(name[i], name[i + 1]) == 1) {

char temp[size];

strcpy(temp, name[i]);

strcpy(name[i], name[i + 1]);

strcpy(name[i + 1], temp);

strcpy(temp, tel[i]);

strcpy(tel[i], tel[i + 1]);

strcpy(tel[i + 1], temp);

b = true;

}

}

} while (b);

cout << "Відсортовано" << endl;

break;

}

cout << "Меню:\n";

cout << "Додати iм'я та номер телефону - |1|, видалення по iменi - |2|, переглянути телефонний довiдник - |3|, сортування по прiзвищу - |4|, вихiд - |0|\n";

cin >> a;

}

return 0;

}

Завдання 4

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<string.h>

#include <iostream>

#include <stdio.h>

#include <stdlib.h>

using namespace std;

#define size 100

int main()

{

setlocale(0, "");

char str[size];

bool b = true;

cin.getline(str, size);

for (int i = 0; i < strlen(str); i++)

{

if (str[i] < '(' || str[i]>'9')

b = false;

}

if (b)

{

char chisla[size], a;

int k = 0, b, index;

for (int i = 0; i < strlen(str); i++)

{

if (str[i] >= '0' && str[i] <= '9')

{

chisla[k] = str[i];

chisla[k + 1] = '\0';

k++;

}

else if (str[i] >= '\*' && str[i] <= '/')

{

a = str[i];

index = i+1;

break;

}

}

b = atoi(chisla);

chisla[0] = '\0';

k = 0;

for (int i = index; i < strlen(str); i++)

{

if (str[i] >= '0' && str[i] <= '9')

{

chisla[k] = str[i];

chisla[k + 1] = '\0';

k++;

}

}

index = atoi(chisla);

switch (a)

{

case '\*':

cout << str;

cout <<"="<< (b \* index);

break;

case '+':

cout << str;

cout << "=" << (b + index);

break;

case '-':

cout << str;

cout << "=" << (b - index);

break;

case '/':

cout << str;

cout << "=" << ((float)b / index);

break;

}

}

else

cout << "Вираз некоректний!\n";

return 0;

}

–

#include<string.h>

#include <iostream>

#include <stdio.h>

using namespace std;

#define s 1000

int char\_To\_Int(const char\* str, unsigned int& start) {

int len = 0;

int chislo = 0;

for (int i = start; '0' <= str[i] && str[i] <= '9' && i <= strlen(str); i++, len++);

for (start; len-- > 0 && start <= strlen(str); chislo += (int)(str[start++] - '0') \* pow(10, len));

return chislo;

}

void Add(int\*& arr, int& size, const int value) {

int\* newArr = new int[size + 1];

for (int i = 0; i < size; i++) {

newArr[i] = arr[i];

}

newArr[size++] = value;

if (arr != nullptr) {

delete[] arr;

}

arr = newArr;

}

void Delete(int\*& arr, int& size, const int index) {

int\* newArr = new int[size - 1];

int i;

for (i = 0; i < index; i++) {

newArr[i] = arr[i];

}

for (i = index + 1; i < size; i++) {

newArr[i - 1] = arr[i];

}

delete arr;

arr = newArr;

size--;

}

int DyskaPriority(const char\* str, int endPosition) {

unsigned short countOfOpenBracket = 0;

unsigned short countOfCloseBracket = 0;

for (int i = 0; i < endPosition; i++) {

countOfOpenBracket += (str[i] == '(');

countOfCloseBracket += (str[i] == ')');

}

return (10 \* (countOfOpenBracket - countOfCloseBracket));

}

int OtrimatPrior(int a) {

switch (a % 10) {

case 1: // +

return 1;

case 2: // -

return 1;

case 3: // \*

return 2;

case 4: // /

return 2;

case 5: // ^

return 3;

default:

return 0;

}

}

void PropyskDyska(const char\* str, unsigned int& position) {

while (str[position] == '(')

position++;

while (str[position] == ')')

position++;

}

int Calculator(char\* str) {

int\* arr = nullptr;

int size = 0;

unsigned int position = 0;

int\* arr2 = nullptr;

int size2 = 0;

for (int i = 0; str[i] != '\0'; i++) {

if (!(('0' <= str[i] && str[i] <= '9' || str[i] == '+' || str[i] == '-' || str[i] == '\*' || str[i] == '/' || str[i] == '^' || str[i] == '(' || str[i] == ')') )) {

cout << "Incorect!" << endl;

return -1;

}

}

do {

PropyskDyska(str, position);

Add(arr, size, char\_To\_Int(str, position));

PropyskDyska(str, position);

switch (str[position++]) {

case '+':

Add(arr2, size2, 1 + DyskaPriority(str, position));

break;

case '-':

Add(arr2, size2, 2 + DyskaPriority(str, position));

break;

case '\*':

Add(arr2, size2, 3 + DyskaPriority(str, position));

break;

case '/':

Add(arr2, size2, 4 + DyskaPriority(str, position));

break;

case '^':

Add(arr2, size2, 5 + DyskaPriority(str, position));

break;

default:

Add(arr2, size2, 0);

}

} while ((arr2[size2 - 1] != 0) && position < strlen(str));

while (arr2[0] != 0) {

int maxPriorytyIndex = 0;

for (int i = 1; i < size2; i++) {

if (arr2[i] / 10 > arr2[maxPriorytyIndex] / 10) {

maxPriorytyIndex = i;

}

else if (arr2[i] / 10 == arr2[maxPriorytyIndex] / 10) {

if (OtrimatPrior(arr2[i]) > (arr2[maxPriorytyIndex])) {

maxPriorytyIndex = i;

}

}

}

switch (arr2[maxPriorytyIndex] % 10) {

case 1:

arr[maxPriorytyIndex] += arr[maxPriorytyIndex + 1];

break;

case 2:

arr[maxPriorytyIndex] -= arr[maxPriorytyIndex + 1];

break;

case 3:

arr[maxPriorytyIndex] \*= arr[maxPriorytyIndex + 1];

break;

case 4:

arr[maxPriorytyIndex] /= arr[maxPriorytyIndex + 1];

break;

case 5:

arr[maxPriorytyIndex] = pow(arr[maxPriorytyIndex], arr[maxPriorytyIndex + 1]);

}

Delete(arr, size, maxPriorytyIndex + 1);

Delete(arr2, size2, maxPriorytyIndex);

}

return arr[0];

}

int main()

{

char str[s];

cin.getline(str,s);

cout<<"="<<Calculator(str);

return 0;

}