#include <string.h>

// String - Xarakterler toplusudur

// String ozu arxada heapde yaradir

// char\* yaradir ve datalari silir

string text = "Hello world!";

int main()

{

string text = "";

text += "Apple";

text += "Apple";

text += "Apple";

cout << text << endl;

cout << text.size() << endl;

cout << text.length() << endl;

cout << text.capacity() << endl;

text += "Apple";

cout << text << endl;

cout << text.size() << endl;

cout << text.length() << endl;

cout << text.capacity() << endl;

cout << text.max\_size() << endl;

return 0;

}

OUTPUT:

AppleAppleApple

15

15

15

AppleAppleAppleApple

20

20

31

2147483647

// String her defe RAMA muraciet etmediyinden char massivinden (char\*) daha suretlidir

X) text.size() - xarakterlerin sayini qaytarir;uzunlugu qaytarir;

X) text.length() - uzunlugu qaytarir;xarakterlerin sayini qaytarir;

X) text.capacity() - minimal olaraq ilk tutumu 15 olur (string bos bele olarsa 15 yer ayiririr); 15i kecdikde tutum artir; bacardigi qeder elave yer ayirir; bir de yer gele biler,yeniden yer axtarmasin deye;

capacity vectorda 2 qat artir; lakin burada her defe 16-16 artir; (ilk olaraq 15)

X) text.max\_size() - bir stringe maximum yazila bilecek xarakter sayidir (2147483647); daha cox lazimdirsa string massivinden istifade edirik;

int main()

{

string s(50, '\*'); // 50 deneli ulduz

cout << s << endl;

cout << s.capacity() << endl;

s.resize(10);

s.shrink\_to\_fit();

cout << s << endl;

cout << s.capacity() << endl;

return 0;

}

OUTPUT:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

63

\*\*\*\*\*\*\*\*\*\*

15

X) s.resize(x) - xarakterlerin sayini x - e azaldir;

X) s.shrink\_to\_fit() - umumi capcityden istifade olunmayan yerleri atir, temizleyir

int main()

{

string text = "Apple";

cout << text.capacity() << endl;

text.reserve(1000);

cout << text.capacity() << endl;

return 0;

}

OUTPUT:

15

1007

X) text.reserve(x) - x ve ya daha cox yer ayirir, reserve edir, capacity-ni deyisir

int main()

{

string text = "Happy programmers day";

cout << text << endl;

text.clear();

if (text.empty())

{

cout << "No Text" << endl;

}

else

{

cout << text << endl;

}

return 0;

}

OUTPUT:

Happy programmers day

No Text

X) text.clear() - stringi silir, temizleyir;

X) text.empty() - stringin bos olub olmadigini gosterir (bool qaytarir) (if - string a = ""; - true else - false)

int main()

{

string text = "Hi all";

cout << text << endl;

text += " , bye bye";

cout << text << endl;

text.append("Salam millet");

cout << text << endl;

return 0;

}

OUTPUT:

Hi all

Hi all , bye bye

Hi all , bye byeSalam millet

X) text.append() - text += Salam ile eynidir;

int main()

{

string name = "Elvin";

string surname = "Camalzade";

cout << name + " " + surname << endl;

return 0;

}

OUTPUT:

Elvin Camalzade

int main()

{

string text = "Happy";

cout << text[0] << endl;

cout << text.at(0) << endl;

text[0] = 'Z';

cout << text << endl;

for (int x = 0; x < text.size(); x++)

{

cout << text[x] << endl;

}

return 0;

}

OUTPUT:

H

H

Zappy

Z

a

p

p

y

X) String - xarakterler toplusu - arraydir - index anlayisi var;

X) text[x] ve text.at(x) - eynidir;

X) text[x] = 'Z'; x indexdeki elementi deyisir

int main()

{

string text = "Happy";

cout << text.front() << endl;

text.front() = 'P';

cout << text.front() << endl;

cout << text.back() << endl;

text.back() = 'i';

cout << text.back() << endl;

cout << text << endl;

return 0;

}

OUTPUT:

H

P

y

i

Pappi

X) text.front - ilk elementin referansini qaytarir

X) text.front() = 'P'; - ilk elementi deyisdi

X) text.back() - son elementin referansini qaytarir

X) text.back() = 'i'; - son elementi deyisdi

int main()

{

string text = "Happy programmers day";

cout << text.substr(6,10) << endl;

cout << text << endl;

return 0;

}

OUTPUT:

programmer

Happy programmers day

X) text.substr(x,y); textin x indexinden y qabaga olan hisseni qaytarir

int main()

{

//insert ile - iteratorla(agilli class)isleyir - her nov container-le nece islemek lazim oldugunu bilir (ilk elementin adresi)

string text = "Happy programmers day";

cout << text << endl;

text.insert(text.begin(), 'E');

cout << text << endl;

text.insert(text.begin() + 5, 'E');

cout << text << endl;

text.insert(text.end() - 1, 'A');

cout << text << endl;

return 0;

}

OUTPUT:

Happy programmers day

EHappy programmers day

EHappEy programmers day

EHappEy programmers daAy

X) text.insert(text.begin() ve ya text.end() += 4, symbol or letter);

int main()

{

string text = "SALAM SALAM";

text.pop\_back();

cout << text << endl;

text.push\_back('Z');

cout << text << endl;

return 0;

}

OUTPUT:

SALAM SALA

SALAM SALAZ

X) text.pop\_back() - sonuncu elementi silir;

X) text.push\_back(const char x) - sona x - i artirir

int main()

{

string text1 = "salam";

string text2 = "Salam";

cout << boolalpha << (text1 == text2) << endl;

cout << boolalpha << text1.compare(text2) << endl;

return 0;

}

OUTPUT:

false

1

X) text1 == text2 - bir textin digerine beraber olmasini, eyni olmasi yoxlayir; Ideal yoxlama budur;

X) text1.compare(text2) - ascii xarakterlerinin cemine gore yoxlayir;0 - dirsa sozler beraberdir, eynidir

int main()

{

string text;

text = "Salam";

cout << text << endl;

string text2;

text2.assign("Salam");

cout << text2 << endl;

return 0;

}

OUTPUT:

Salam

Salam

X) text = smth ve .assign() eynidir

int main()

{

string text = "Happy programmers day";

cout << text.find('j') << endl; // Olmayanda ekrana boyuk reqem qaytarir

cout << text.find('H') << endl; // Olanda ekrana ilk elementin indexini tapir

cout << text.rfind('j') << endl;

cout << text.rfind('y') << endl;

return 0;

}

OUTPUT:

X) .rfind() stringin sagindan isleyir

X) String input - cin >> text; stringin ilk bosluguna qeder - ilk qirilma noqtesine qeder - goturur

X) #include <string> - getline(cin, text); normal input