Fundamentele Programarii

Caractere si stringuri

Caractere si stringuri

- Functii folosite pentru caractere

- Stringuri in stilul C (secvente de caractere)
- Lucrul cu stringuri in stilul C

- Stringuri in stilul C++ (obiecte)
- Lucrul cu stringuri in stilul C++

Functii folosite pentru caractere

- **# include <cctype>** (in noile versiuni nu mai e nevoie)
  nume_functie (nume_caracter)

- Functii pentru testarea caracterelor
- Functii pentru conversia caracterelor

Functii folosite pentru testarea caracterelor

isalpha (c)	True daca c este o litera
isalnum (c)	True daca c este o litera sau o cifra
isdigit(c)	True daca c este o cifra
islower (c)	True daca c este o litera mica
isprint (c)	True daca c este un caracter printabil
ispunct (c)	True daca este un semn de punctuatie
isupper (c)	True daca c este o litera mre
isspace (c)	True daca c este spatiu gol

Functii folosite pentru conversia caracterelor

tolower (c)	Returneaza codul ascii al literei mici corespunzatoare lui c
toupper (c)	Returneaza codul ascii al literei mari corespunzatoare lui c

- Daca vrem sa apara litera mica, respectiv litera mare trebuie sa facem conversia la char

(char) tolower(c)
(char) toupper(c)

https://en.cppreference.com/w/cpp/language/ascii

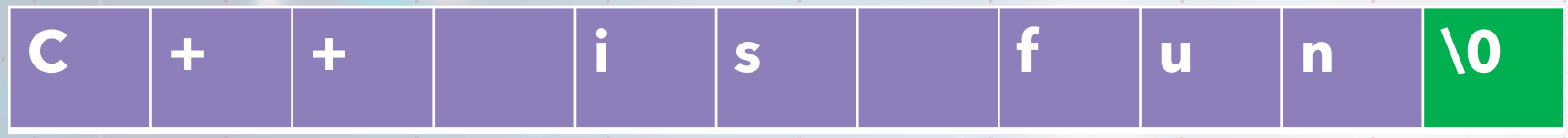
Stringuri in stilul C

- Secvente de caractere
  - Pastrate impreuna, in mod continuu, in memorie
  - Implementate ca un tablou de caractere
  - Terminate cu caracterul `null` (caracterul null are valoarea literala zero ‘\0’)
  - Caracterul `null` este un caracter de control care nu este printabil si care are valoarea `0`

- Stringuri literale
  - Secvente de caractere declarate cu ghilimele duble (exemplu, “`Simona`”)
  - Constante
  - Se termina cu caracterul `null` (chiar daca nu il precizam noi explicit, C++ il pune automat)

Stringuri in stilul C

char text [] {“C++ is fun”};



- Tablou de caractere
- Aceste caractere sunt retinute in mod continuu in memorie
- Putem accesa fiecare caracter asa cum procedam la tablouri
- Are 10 caractere, dar compilatorul aloca 11 caractere pentru tablou, pentru ca are nevoie de spatiu pentru caracterul null de la sfarsit

Stringuri in stilul C – declararea varibilelor

```
char nume [] {“Simona”};
```

- Aloca 7 caractere pentru acest tablou
- Dimensiunea tabloului este fixa
- `nume[6] =‘E’;` //problema pentru ca stringul nu se mai termina cu caracterul `null`, dar totusi nu se generaza o eroare sau o avertizare

```
cout<<nume;
```

SimonaE╠╠╠╠╠╠╠╠

Stringuri in stilul C – declararea varibilelor

```
char nume [9] {“Simona”};
```

- `nume[6]='E';` //ok –stringul se termina cu caracterul null
  `cout<<nume;` SimonaE

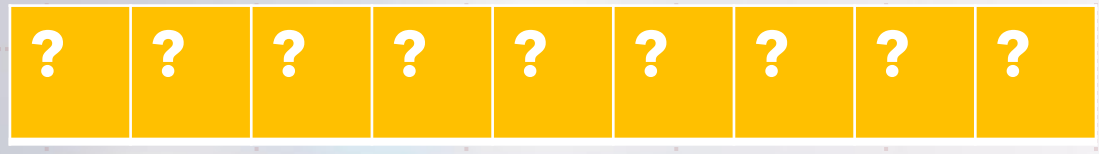
Stringuri in stilul C – declararea varibilelor

```
char nume [9] {“Simona”};
```

- `nume[7]='E';` //ok –stringul se termina cu caracterul null
  `cout<<nume;` Simona

Stringuri in stilul C – declararea varibilelor

```
char nume [9] ;
```



- Se aloca memorie pentru 9 caractere; tabloul va avea ‘efectiv’ 8 caractere, dar nu este initializat
- Poate fi problematic, deoarece compilatorul se asteapta ca la finalul stringului sa intalneasca caracterul `null`.
- Daca il afisam nu putem sti ce se va genera (nu stim ce si cate caractere vor fi afisate pentru ca nu stim cand se va intalni caracterul `null`)

- `nume= “Simona”;` //eroare

- **`strcpy`** `(nume, “Simona”);` // functia **strcpy** copiaza cuvantul *Simona* in tabloul *nume*

- `cin>>nume;`
- `cout<<nume;` *//Daca tastam “CursInformatica”, ce se intampla?*

CursInformatica

```
cin.getline(nume,9);
cout<<nume;
```

CursInfo

Functii folositoare pentru stringuri in stilul C

- In versiunile mai vechi trebuia sa includem: #include<cstring>

- **Copiere**: `strcpy ( text1 , text2 );` // copiaza valoarea lui `text2` la adresa lui `text1`; atentie la dimensiunea lui `text1`!
- Copierea unui numar limitat de caractere: `strncpy (text1, text2, n);` // copiaza `n` caractere din continutul lui `text2` la adresa lui `text1` si pune la final '`\0`'; atentie la dimensiunea lui `text1`!

```
char text1[5];
char text2[] = "Testare";
strcpy(text1,text2); //Runtime error!
strncpy(text1, text2, 4); // Copiază primele 4 caractere
cout << text1; // Test
```

Functii folositoare pentru stringuri in stilul C

- **Concatenare:** `strcat ( text1 , text2 );` // concateneaza valorea lui `text2` la valoarea lui `text1` si o retine la adresa lui `text1`; atentie la dimensiunea lui `text1`!
- Concatenarea unui numar limitat de caractere: `strncat (text1, text2, n);` // concateneaza n caractere din continutul lui `text2` la adresa lui `text1` si pune la final `'\0'`; atentie la dimensiunea lui `text1`!

- **Lungime:** `strlen (text);` // returneaza o variabia de tip **size_t** (`unsigned int` sau `unsigned long`, in functie de compilator) care numara cate caractere sunt in text pana se intalneste caracterul `'\0'`.

- Atentie! Intotdeauna trebuie sa avem spatiu in tablou pentru caracterul `null`!

Presupunem ca avem un string in stilul C si vrem sa ii determinam lungimea sa. Incepem sa numaram caracterele incepand cu primul caracter pana se intalneste caracterul `null`. Daca caracterul `null` lipseste noi vom continua sa numaram, deci evident nu se va obtine un rezultat corect.

Functii folositoare pentru stringuri in stilul C

- **Comparare:** `strcmp( text1 , text2);` // compara lexicografic `text1` si `text2` si returneaza `0` daca cele doua strigniuri coincid, `1 (>0)` daca primul string este mai mare lexicografic decat al doilea si `-1(<0)` daca al doilea string este mai mare lexicografic decat primul.
- Comparare limitata: `strncmp (text1,text2, n);` // compara lexicografic doar primele `n` caractere din `text1` si `text2`
- **Gasirea unui caracter in sir:** `strchr(text, 'x');` // cauta prima aparitie a unui caracter intr-un sir; returneaza un pointer la pozitia caracterului gasit sau `nullptr` daca nu este gasit

```
char text[] = "Salut";
char *pos = strchr(text, 'a');
if (pos)
    cout << "Caracterul 'a' găsit la poziția: " << (pos - text) << endl;
```

1

Functii folositoare pentru stringuri in stilul C

- **Gasirea ultimei aparitii a unui caracter in sir:** `strrchr(text, ‘x’);` // cauta ultima aparitie a unui caracter intr-un sir; returneaza un pointer la pozitia caracterului gasit sau `nullptr` daca nu este gasit

```
char text[] = “Matematica";
char *pos = strrchr(text, 'a');
if (pos)
    cout << "Caracterul 'a' găsit la poziția: " << (pos - text) << endl;
```

9

- **Gasirea unui sir intr-un alt sir:** `strstr(text1, text2);`// cauta prima aparitie a unui sir in alt sir; returneaza un pointer la inceputul subsirului gasit sau `nullptr` daca nu este gasit

```
char text1[] = "Bună ziua";
char text2[] = "ziua";
char *pos = strstr(text1, text2);
if (pos)
    std::cout << "Subșirul găsit la poziția: " << (pos - str) << std::endl;
```

5

Functii folositoare pentru stringuri in stilul C

```
char text[80];
cout << text<<endl;
strcpy(text, "Buna "); //copierea; "Buna" este un literal string si deci se termina cu \0
cout << text << endl;
strcat(text, "ziua"); //concatenarea
cout << text << endl;
cout <<"Lungimea stringului este ";
cout << strlen(text); //lungimea stringului
cout << endl;
cout<<strcmp(text, "Buna seara"); //comparare a doua stringuri; returneaza
//strcmp(text1,text2) returneaza 0 daca text1 == text2

//strcmp(text1,text2) returneaza 1 daca primul caracter din text1 diferit de caracterul
//din text2 este mai mare lexicografic(conform codului Ascii)

//strcmp(text1,text2) returneaza -1 daca primul caracter din text1 diferit de caracterul
//din text2 este mai mic lexicografic(conform codului Ascii)
```

Buna

Buna ziua

9

1

https://en.cppreference.com/w/cpp/language/ascii

Afisarea stringurilor in stilul C

```
char sursa[]{ "Acesta este un exemplu" };
cout << "====================" << endl;
cout << "Prima afisare" << endl;
cout << sursa;
cout << endl;

cout << "====================" << endl;
cout << "A doua afisare" << endl;
for (int i{ 0 }; sursa[i]; i++) {
cout << sursa[i];
}
cout << endl;

cout << "====================" << endl;
cout << "A treia afisare" << endl;
for (char c: sursa) {
cout << c;
}
cout << endl << endl;
```

Citirea stringurilor in stilul C

```
char sursa[100]{};
cout << "Introduceti un text: ";
cin >> sursa;
cout << "Textul introdus este: " << sursa;
```

Acesta este un curs de informatica.

Acesta

```
char sursa[100]{};
cout << "Introduceti un text: ";
cin.get(sursa,100);
cout << "Textul introdus este: " << sursa<<endl<<endl;
```

Acesta este un curs de informatica.

Acesta este un curs de informatica.

```
char sursa[100]{};
cout << "Introduceti un text: ";
cin.getline(sursa, 100);
cout << "Textul introdus este: " << sursa<<endl;
```

Acesta este un curs de informatica.

Acesta este un curs de informatica.

Se citesc maxim 100 de caractere, pana se intalneste o linie noua '\n';
cin.get() pastreaza '\n' (ramane '\n' in buffer), iar cin.getline() nu pastreaza '\n' (elimina '\n' din buffer)

Citirea caracterelor si a stringurilor in stilul C

Vrem sa citim de la tastatura un caracter si un string in stilul C (o propozitie).

```
char litera;
cout << "Introduceti un carcater: ";
cin >> litera;
char sursa[100]{};
cout << "Introduceti un text: ";
cin.getline(sursa, 100);
cout<<“Sfarsit”;
```

Daca in consola vom introduce un caracter, apoi apasam ENTER, nu ni se va permite sa introducem textul pe care vrem sa il retinem in tabloul sursa, ci se va fisa direct “Sfarsit”. De ce?

`cin >> litera;` citeste un caracter, dar nu elimina/foloseste caracterul de linie noua `‘\n’` care ramane in buffer-ul de intrare (adica dupa ce utilizatorul apasa ENTER)

`cin.getline(sursa, 100);` citeste tot ce se afla in buffer pana la `‘\n’` cand se atinge dimensiunea maxima.

Deoarece caracterul `‘\n’` a ramas in buffer de la primul `cin`, functia `getline` considera ca
Sirul de intrare este gol si citeste doar acel `‘\n’`. Astfel, nu putem citi sirul dorit in `sursa` deoarece buffer-ul continea doar `‘\n’`

Cum rezolvam aceasta problema?

Citirea caracterelor si a stringurilor in stilul C

SOLUTIE 1: Adaugam cin.ignore() dupa cin

```
char litera;
cout << "Introduceti un caracter: ";
cin >> litera;
cin.ignore();   // cin.get(); elimina caracterul ‘\n’ ramas in buffer
char sursa[100]{};
cout << "Introduceti un text: ";
cin.getline(sursa, 100);
cout<<“Sfarsit”;
```

SOLUTIE 2: Schimbam ordinea: citim mai intai stringul si apoi caracterul

```
char sursa[100]{};
cout << "Introduceti un text: ";
cin.getline(sursa, 100);
char litera;
cout << "Introduceti un caracter: ";
cin >> litera;
cout<<“Sfarsit”;
```

Citirea caracterelor si a stringurilor in stilul C

Aceasi problema apare si in cazul urmator:

```
char text1[100], text2[100];
cout << "Introduceti primul text: ";
cin.get(text1, 100);
cout << "Introduceti al doilea text: ";
cin.getline(text2, 100);
cout << "Sfarsit";
```

SOLUTIE 1: Adaugam cin.ignore() dupa cin.get(.., ..);
SOLUTIE 2: Schimbam ordinea, folosind prima data cin.getline (.., ..) dupa cin.get(.., ..);
SOLUTIE 3: Folosind doar cin.getline (.., ..).

Exemplu de cod gresit

```
char text[]{ "Bun" };
text[3] = 'a';
cout << text<<endl; Buna╠╠╠╠êvô
cout << strlen(text);    11
```

Vezi programul “Stringuri-C”!

Stringuri in stilul C++

- `std:: string` este o clasa in Standard Template Library
- **`#include<string>`**
- `std namespace`
- Sunt pastrate in mod continuu in memorie
- **Dimensiune dinamica**
- Multe functii si metode utile care ne permit sa manipulam usor stringurile
- **Operatorii familiari** pot fi folositi (`+, =, <, <=, >=, +=, ==, !=,….`)
- Pot fi usor convertiti in stringuri in stilul C, daca se doreste
- Sunt mai siguri pentru ca avem posibilitatea sa facem verificari asupra dimensiunii maxime, astfel incat sa nu obtinem erori

Stringuri in stilul C++

- **Exemple:**

```
string s1;  // stringurile sunt automat initializate cu caracterul null (nu avem “garbage”)
string s2 {“Simona”};                          // Simona
string s3{s2};                                 // Simona
string s4{“Simona”,3};                        // Sim
string s5{s3,0,2};                           // Si     (incepe cu poz 0 si are lungimea 2)
string s6 (3, ‘X’);                                  // XXX
```

Stringuri in stilul C++

- **Operatorul de atribuire**

```
string s1;
s1="C++ este cel mai bun!"; //nu putem face acest lucru cu stringuri in stil C

string s2{"Buna"};
s2=s1;                                  //s2=C++ este cel mai bun!
```

Stringuri in stilul C++

- **Concatenarea**

```
string partea1{“C++”};
string partea2 {“este cel mai bun”};

string propozitie;
propozitie=partea1 + “  ” +partea2 + “ limbaj de programare”; // C++ este cel mai bun limbaj de programare

propozitie=“C++” + “ este puternic”; //ilegal (pentru ca avem doi literali string)
```

Stringuri in stilul C++

- **Accesarea caracterelor se poate face cu *[ ]* sau cu metoda *at* (ca si la vectori)**

```
string s {“Simona”};

cout<< s[0]<<endl;          // S
cout<<s.at (0)<<endl;    // S

s[0]=‘s’;                              // simona
s.at (0) = ‘L’;                     // Limona
```

- Reamintim ca metoda *at* asigura o verificare a dimensiunii maxime, adica daca apelam de exemplu s2.at(10) vom obtine o eroare

Stringuri in stilul C++

- **Accesarea caracterelor unui string in stilul C++**

```
string s {“Simona”};

for (char c: s)
        cout<<c<<endl;
```

```
S
i
m
o
n
a
```

```
string s {“Simona”};

for (int c: s)
        cout<<c<<endl;
```

```
83  //S
105 //i
109 //m
111 //o
110 //n
97  //a
```

Stringuri in stilul C++

- **Afisarea stringurilor**

```
string s1{ "Acesta este un test" };
cout << "Prima afisare:" << endl;
cout<<s1 << endl;

cout << "======================" << endl;
cout << "A doua afisare:" << endl;
for (size_t i{ 0 };i < s1.length();i++)
          cout << s1.at(i);
cout << endl;

cout << "======================" << endl;
cout << "A treia afisare:" << endl;
for (size_t i{ 0 };i < s1.length();i++)
          cout << s1[i];
cout << endl;

cout << "======================" << endl;
cout << "A patra afisare:" << endl;
for (char c:s1)
          cout << c;
cout << endl;
```

```
cout << "======================" << endl;
cout << "A cincea afisare:" << endl;
size_t i{ 0 };
while (i < s1.length()) {
          cout << s1.at(i);
          i++;
}
cout << endl;

cout << "======================" << endl;
cout << "A sasea afisare:" << endl;
size_t j{ 0 };
do{
          cout << s1.at(j);
          j++;
} while (j < s1.length());
cout << endl;

cout << "======================" << endl;
cout << "A saptea afisare:" << endl;
for (size_t j{ 0 }; s1[j]; j++)
          cout << s1[j];
```

Stringuri in stilul C++

- **Compararea ==, !=, >, >=, <, <=**
- Obiectele sunt comparate lexicografic caracter cu caracter
- Putem compara:

----- doua stringuri C++;
----- un string C++ si un string literal;
----- un string C++ si un string C

Stringuri in stilul C++

- **Compararea ==, !=, >, >=, <, <=**
- **Exemple:**

```
string s1{ “Mar”};
string s2{ “Banana”};
string s3{ “Kiwi”};
string s4{ “mar”};
string s5{ s1};   //Mar

cout << (s1==s5)<<endl;
cout << (s1==s2)<<endl;
cout << (s1!=s5)<<endl;
cout << (s1<=s2)<<endl;
cout << (s2>s1)<<endl;
cout<< (s4>s5)<<endl;
cout << (s1<=“Mar”)<<endl;
char s6[]{“Portocala”};
cout << (s6>=s3)<<endl;
```

1

0

0

0

0

1

1

1

Stringuri in stilul C++

- **Substringuri**

- Trebuie sa includem directiva **#include<string>**
- Metoda **substr(..,..)** extrage un substring dintr-un string
- Sintaxa:

nume_string.substr (index_start, lungime);

```
string s1{ “Acesta este un test”};

cout << s1.substr(0,5)<<endl;        //Acest
cout << s1.substr(5,3) <<endl;       //a e
cout << s1.substr(10,4)<<endl;      //e un
cout << s1.substr(16, 6) << endl;    //est  (daca lungime substringului depaseste lungimea
stringului se vor afisa doar cate caractere are stringul initial)
```

Stringuri in stilul C++

- **Cautare**

- Metoda **find()** returneaza indexul unde se gaseste (prima data) un anumit substring intr-un string
- Sintaxa:

nume_string.find (substringul_cautat);

```
string s1{ “Acesta este un test”};

cout << s1.find(“Acest”)<<endl;          //0
cout << s1.find (“est”) <<endl;          //2
cout << s1.find(“test”)<<endl;          // 15
cout << s1.find(“est”,3) << endl;    //7 (se cauta “est” de la pozitia 3 incolo)
cout << s1.find(‘t’) << endl;    //4
cout << s1.find(“XXX”) << endl;    // 4294967295 (garbage)
```

Stringuri in stilul C++

- **Stergerea caracterelor**

- Trebuie sa includem directiva **#include<string>**
- Metoda **erase()** sterge un anumit substring intr-un string
- Sintaxa: **nume_string.erase (index_start, lungime);**

- Metoda **clear()** sterge toate carcterele stringului, acesta devenind un string gol
- Sintaxa: **nume_string.clear ();**

```
string s1{ “Acesta este un test”};

cout << s1.erase(0, 5) << endl;        // a este un test
cout << s1.erase(5, 2) << endl;       //a estun test
cout << s1.erase(8, 7) << endl;       //a estun
s1.clear();
cout << s1;// string gol
```

Stringuri in stilul C++

- **Alte metode folositoare**

```
string s1{ "Acesta este un test" };
cout << s1.length() << endl;     //19
s1 += ". El este foarte interesant.";
cout << s1 << endl;   //Acesta este un test. El este foarte interesant.
```

- **Si altele...**

Stringuri in stilul C++

- **Citirea unui string: cin>> si getline()**

```
string s2;

cin >> s2;//citeste pana se intalneste primul spatiu
cout << s2 << endl;

getline(cin, s2); //citeste intreaga linie pana intalneste \n
cout << s2 << endl;

getline(cin, s2, 'x'); //citeste pana se intalneste 'x'; daca nu se intalneste 'x' nu se
afiseaza nimic daca apasam tasta enter; se asteapta sa introducem in continuare de la
tastatura; putem scrie pe mai multe randuri si abia dupa ce folosim caracterul 'x' va fi
afisat tot textul pana la 'x'
cout << s2 << endl;
```

Acesta este numele meu.

Acesta

Acesta este numele meu.

Acesta este numele meu.

Acesta este numele meu.
Inca nu am scris caracterul respectiv.
Acesta este un e

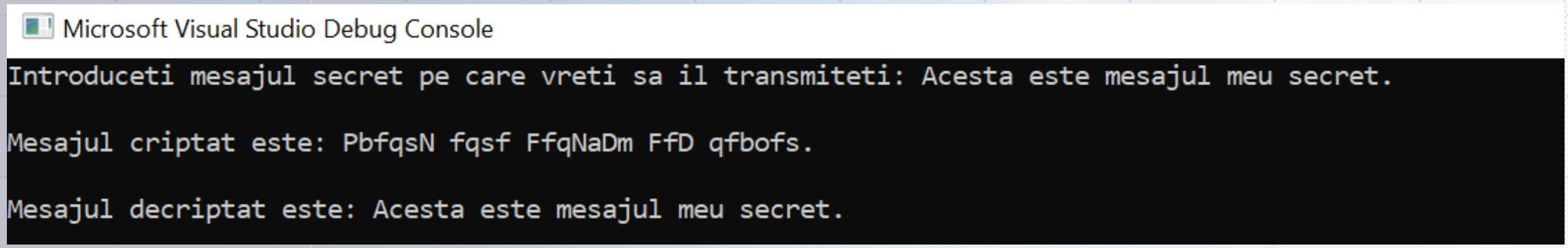
Acesta este numele meu.
Inca nu am scris caracterul respectiv.
Acesta este un exemplu.

Vezi programul “StringuriCPlusPlus”!

Stringuri in stilul C++

- **Problema:** Realizati un program care sa cripteze si sa decripteze un mesaj introdus de utilizator.

```
string alfabet {"abcdefghijklmonpqrstuvxywzABCDEFGHIJKLMNOPQRSTUVXYWZ"};
string cheie{"NcbgfdHJjaZmFIlBpoqsDteurvPxywLCzAihKTRYESnMOUQVXWkG"};
```



https://test-master.space