Universidade Federal do Ceará – Campus Quixadá
Maratona de Programação 2012 – Notebook

Equipes:
#include<GambiarrasWladimir.h>
Why so C_rious?

17 de agosto de 2012

Índice

1. Algoritmo de Euclides: MDC – Máximo Divisor Comum

```
int Euclides(int a, int b){
    if(b==0)
       return a;
    else
       return Euclides(b,a%b);
}

int MDC(int a, int b){
    int c;
    while(b!=0){
       c = a%b;
       a=b;
       b=c;
    }
    return a;
}
```

2. MMC – Mínimo Múltiplo Comum

```
int MMC(int a, int b){
    // mmc(a,b)*mdc(a,b)=a*b
    return (a*b)/MDC(a,b);
}
```

3. Exponenciação na base 2 com deslocamento de bits

```
int exp=10;
printf("%d\n",(1 << exp));
// 2^10 = 1024, exp máximo=30
```

4. Compilar um arquivo C no Linux e linkar a math

```
C =>  gcc test.c -o test -lm
C => gcc -lm test.c -o teste
C++ => g++ -lm teste.cpp -o teste
```

5. Teste de Primalidade

```
#include <math.h>

int Primo(int n){
    int i;
    int raiz = sqrt(n);
    for(i=2; i<=raiz; i++){
       if( n%i == 0 )
            return 0;
    }
    return 1;
}
//Obs.: linkar math no Linux.
```

6. Distância entre dois pontos no plano

```
#include <math.h>
#define POW(x) ((x)*(x))

typedef struct{
    int x,y;
}Ponto;

double Distancia(Ponto p1, Ponto p2){
      return  sqrt( POW(p1.x - p2.x) +
                    POW(p1.y - p2.y) );
}
```

7. Fatorial

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
11! = 39916800
12! = 479001600   [limite do int]
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
17! = 355687428096000
18! = 6402373705728000
19! = 121645100408832000
20! = 2432902008176640000 [limite do long long int]
```

```
typedef long long int Big;

Big Fatorial(int n){
    int i;
    Big fat;
    for(i=1,fat=1; i<=n; i++){
        fat*= i;
    }
    return fat;
}

int Fat(int n){
    if(n==0)
       return 1;
    else
       return n*Fat(n-1);
}
```

8. Algoritmo de Preenchimento de Regiões

```
#include <stdio.h>
#include <string.h>

#define MAX 200
#define BLACK 1
#define WHITE 0
```

```
int mat[MAX+2][MAX+2];
memset(mat,WHITE,sizeof(mat));

//borda linha
for(i=0; i<n+2; i++){
    mat[i][0]=BLACK;
    mat[i][m+1]=BLACK;
}

//borda coluna
for(j=0; j<m+2; j++){
    mat[0][j]=BLACK;
    mat[n+1][j]=BLACK;
}

void Exibir(int mat[][MAX+2]){
     int i,j;
     for(i=0; i<n+2; i++){
        for(j=0; j<m+2; j++){
             printf("%d ",mat[i][j]);
        }
         printf("\n");
     }
}

void Pintar(int mat[][MAX+2], int x, int y){
     if( (x>=1 && x<=n) &&
         (y>=1 && y<=m) ){
       if(mat[x][y]==WHITE){
            count++;
             mat[x][y]=BLACK;

             Pintar(mat,x+1,y);
             Pintar(mat,x+1,y+1);

             Pintar(mat,x-1,y);
             Pintar(mat,x+1,y-1);

             Pintar(mat,x,y+1);
             Pintar(mat,x-1,y+1);

             Pintar(mat,x,y-1);
             Pintar(mat,x-1,y-1);
         }
     }
}
```

9. Calcular Tempo de Execução

```
#include <stdio.h>
#include <time.h>

int main(){
  time_t start, stop;
  double tempo = 0.0;
  start = clock();

  // lógica do programa

  stop = clock();
  tempo = (double) (stop-start)/CLOCKS_PER_SEC;

  printf("Tempo de Execucao: %lf s\n", tempo);
  system("PAUSE");
  return 0;
}
```

10. Exponenciação Rápida

```
int Exp(int a, int b){
    if(b==0)
       return 1;
    else if(b==1)
       return a;
    else{
       int x = Exp(a,b/2);
       if(b%2==0)
          return (x*x);
       else
          return (x*x*a);
    }
}
```

11. Inverter Vetor

```
void Inverter(int n, int v[]){
     int i, aux;
     for(i=0; i<n/2; i++){
          aux = v[i];
          v[i] = v[n-1-i];
          v[n-1-i] = aux;
     }
}
```

12. Converter String para Inteiro

```
int i, soma=0;
char string[11]="0123456789";

for(i=0; string[i]!='\0'; i++)
     soma+= (string[i]-'0');
printf("SOMA: %d\n",soma);  //SOMA: 45
```

13. Sequência de Fibonacci

```
int Fib(int n){
    if(n==0)
       return 0;
    else if(n==1 || n==2)
       return 1;
    else
       return Fib(n-1)+Fib(n-2);
}

void Fib_iterativo(){
    int i;
    int a=0, b=1,c;
    for(i=1; i<=10; i++,a=b,b=c){
        c=a+b;
        printf("%d\n",b);
    }
}
```

14. Busca Binária

```
int BuscaBinaria(int n, int v[], int k){
    int meio, esq, dir;
    esq=0; dir=n-1;
    while(esq <= dir){
         meio=(esq+dir)/2;
         if(v[meio]==k)
             return meio;
         else if(v[meio] < k)
             esq = meio+1;
         else if(v[meio] > k)
             dir = meio-1;
    }
    return -1;
}
```

15. Função qsort

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 6

int Compare(const void *a,const void *b){
  return ( *(int*)a - *(int*)b );
}

int main (){
  int i;
  int vetor[MAX]={40,10,100,90,20,25};

  qsort (vetor, n, sizeof(int), Compare);

  for(i=0; i<6; i++)
     printf("%d ",vetor[i]);

  return 0;
}
```

16. Library cstring(string.h)

16.1 memset: preencher bloco de memória

```
    int vetor[10];
    memset(vetor,0,sizeof(vetor));
```

16.2 strcmp: comparar strings

```
    char S[5]="a";              //0, S=E
    char E[5]="z";              //1, S>E
    printf("%d\n",strcmp(S,E)); //-1, S<E
```

16.3 strcpy: copiar string

```
    char str1[11];
    char str2[]="orange";
    strcpy(str1,str2); //strcmp(str1,str2)==0
```

16.4 strlen: retorna o tamanho da string

```
    char S[]="orange";
    printf("%d\n",strlen(S));  //6
```

17. Matriz de Adjacência

```
#include <stdio.h>
#include <string.h>

#define MAX 101

int grafo[MAX][MAX];

int main(){
    memset(grafo,0,sizeof(grafo));
    int i,j, a, b, arestas, nos;
    scanf("%d %d",&nos, &arestas);

    for(i=1; i<=arestas; i++){
         scanf("%d %d",&a,&b);
         grafo[a][b]=grafo[b][a]=1;
    }

    for(i=1; i<=nos; i++){
        for(j=1; j<=nos; j++){
            printf("%d ",grafo[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

18. Número de Inversões em Vetor

```
// Um algoritmo O(n log n)
#include <stdio.h>

typedef long long int Big;

Big merge_count(int A[], int B[], int p,
                              int q,int r){
 int i,j,k;
 long long int c;

 for(i=p;i<=q;i++)
    B[i] = A[i];

 for(j=q+1;j<=r;j++)
    B[r+q+1-j]=A[j];

 i = p;
 j = r;
 c = 0;

 for(k=p;k<=r;k++){
   if(B[i] <= B[j]){
     A[k] = B[i];
     i = i+1;
   }else{
     A[k] = B[j];
     j = j-1;
     c = c + (q-i+1);
   }
 }

 return c;
}
```

```
Big sort_count(int A[], int B[], int i,int j){
     int q;
     if(i >=j )
        return 0;
     else{
         q = (i+j)/2;
         return sort_count(A, B, i, q) +
                sort_count(A, B, q+1, j) +
                merge_count(A, B, i, q, j);
     }
}

int main(){
    int i, n;
    int v[100000];
    int aux[100000];
    Big res;

    scanf("%d",&n);
    for(i=0; i<n; i++){
          scanf("%d",&v[i]);
    }

    res=0;
    res = sort_count(v,aux,0,n-1);

    printf("%lld\n",res);

    getch();
    return 0;
}
```

19. Otimizar Casos de Teste

Onde no arquivo entrada.txt contém todas às entradas e no final da execução o resultado do processamento se encontrará no arquivo saida.txt.

Windows:
(nome_arquivo.exe) < entrada.txt > saida.txt

Linux:
./(nome_arquivo_compilado) < entrada.in > saida.out

Obs.: Para realizar isso, será preciso abrir o terminal do windows/linux (prompt ou cmd), e em seguida, navegar até a pasta onde se encontrar o arquivo (nome_arquivo.exe)/(nome_arquivo_compilado).

20. Subset-sum

Problema Subset-sum: [1]
Dados números naturais p1, p2, ... , pn e c, decidir se existe um subconjunto X de {1,2, ... , n} tal que p(X) = c.

Diremos que os números *p1*, *p2*, ..., *pn* são os pesos e *c* é a capacidade do problema. (A ordem em que os pesos são dados é, obviamente, irrelevante.) O problema só admite duas soluções, "sim" (1) e "não" (0).

Ex.:
p = {30, 40, 10, 15, 10, 60, 54} e *c = 55* tem solução "sim" (1) pois o conjunto *{40,15}* tem a propriedade. Já o conjunto *{30, 15, 10}* também tem a propriedade desejada.

20.1 Estrutura recursiva do problema

```
int subsetsum(int p[], int n, int c){
    int s;

    if(n==0){
         if(c==0)
             return 1;
         else
             return 0;
    }else{
         s = subsetsum(p,n-1,c);
         if(s==0 && p[n-1]<=c)
            s = subsetsum(p,n-1,c-p[n-1]);
    }
    return s;
}
```

20.2 Estrutura iterativa do problema

```
#include <string.h>
#define MAX 219 //soma total de coin

int i,j;
int c = 55;
int coin[7] = {30,40,10,15,10,60,54};
int count[MAX+1];

memset(count,0,sizeof(count));

count[0] = 1;
for (i=0;i<7;i++)
    for (j=MAX;j>=coin[i]; j--)
         count[j] |= count[j-coin[i]];

printf("%d\n",count[c]);
```

21. Estrutura de Dados para Conjuntos Disjuntos

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXN 100001

int p[MAXN];
int ordem[MAXN];
int mark[MAXN];
```

```
void make_set(int x){
     p[x] = x;
     ordem[x] = 0;
}
 int find_set(int x){
      if( x!= p[x] )
            p[x] = find_set(p[x]);
      return p[x];
}


int same_componente(int x,int y){
    if(find_set(x)==find_set(y))
       return 1;
    else
       return 0;
}

void link(int x,int y){
      if(ordem[x] > ordem[y]){
            p[y] = x;
      }else{
            p[x] = y;
            if(ordem[x]==ordem[y])
            ordem[y]++;
      }
}


void une(int x,int y){
      link(find_set(x),find_set(y));
}


int main(){
      int i;
      int n, m;
      int a, b;
      int count, aux;

      scanf("%d %d",&n,&m);

      for(i=1; i<=n; i++)
       make_set(i);

      for(i=1; i<=m; i++) {
         scanf("%d %d",&a,&b);
         une(a,b);
    }

      memset(mark,0,sizeof(mark));

     count=0;
     for(i=1; i<=n; i++){

         aux = find_set(i);
         if( mark[aux] == 0 ){
             mark[aux]=1;
             count++;
         }

     }

    printf("%d\n",count);

    return 0;
}
```

22. Troco de Moedas

Dados o preço de uma mercadoria e os valores das moedas disponíveis, calcule o menor número possível de moedas necessário para comprar o produto sem haver troco, ou seja, o menor número de moedas tal que o total seja exatamente o preço da mercadoria. [2]

```
#include <stdio.h>
#define INF ~(1<<31)

int min(int a, int b) {
    return a<b?a:b;
}

int coin[100];
int count[50001];

int main(){
    int i, j;
    int n, m;

    while(1){
       scanf("%d",&m);
       if(m==0)break;
       scanf("%d",&n);

       for(i=0; i<n; i++){
           scanf("%d",&coin[i]);
       }

       for(i=0;i<=m;i++)
           count[i] = INF;

       count[0] = 0;
       for (i=0;i<n;++i)
           for (j=coin[i];j<=m;++j)
              if(count[j-coin[i]] != INF)
                  count[j] = min(count[j],
                         count[j-coin[i]]+1);

    if( count[m] != INF )
       printf("%d\n",count[m]);
    else
       printf("Impossivel\n");


    }

    return 0;
}
```

23. Escalonamento de Intervalos

Dado um conjunto n atividades (k_1, k_2,..., k_n). Onde cada k_i é tem um início s_i e um término f_i. Qual o tamanho do maior subconjunto das n atividades onde não existe choque de intervalos.

Dica: ordenar por f_i.

Complexidade:

- $\theta(n \log n)$ para ordenar por f_i.
- $\theta(n)$ para o resto do algoritmo.
- Total: $\theta(n \log n)$.

```
#include <stdio.h>
#include <stdlib.h>
#include <algorithm>
#include <vector>
#include <utility> //pair, make_pair

using namespace std;

bool Cmp( pair<int,int> i, pair<int,int> j){
        return i.second < j.second;
}

int main(){
    int n, i;
    int a, b;
    int count;
    vector< pair<int,int> > v;

    while(1){
    scanf("%d",&n);
    if( n == 0)break;

    v.clear();
    for(i=0; i<n; i++){
        scanf("%d %d",&a,&b);
        v.push_back(make_pair(a,b));
    }

    sort(v.begin(), v.end(), Cmp);

    count=1;
    for( i=1; i<n; i++){
         if( v[0].second <= v[i].first  ){
              count++;
             v[0] = v[i];
         }

    }
        printf("%d\n",count);
    }

    return 0;
}
```

24. Grafo Bipartido com Lista de Adjacência

Um grafo bipartido é um grafo cujos vértices podem ser divididos em dois conjuntos disjuntos *U* e *V* tais que toda aresta conecta um vértice em *U* a um vértice em *V*. [3]

Um grafo é dito bipartido se e somente se o grafo poder ser colorido por duas cores (branco e preto) de forma que dois vértices vizinhos não tenham mesma cor.

```
#include <stdio.h>
#include <string.h>

#define AZUL 0
#define PRETO 1
#define BRANCO 2

int mat[101][101];
int cor[101];
int d[101];
int n, m;
int bipartido;


void Bipartido(int v, int color){
    int i;
    int viz;
    cor[v] = color;
    for(i=0; i<d[v]; i++){
       viz = mat[v][i];
       if( cor[viz] == AZUL ){
            if( cor[v] == PRETO )
                  Bipartido(viz, BRANCO);
            else if( cor[v] == BRANCO )
                  Bipartido(viz, PRETO);
       }else if( cor[v] == cor[viz] ){
                  bipartido = 0;
                  break;
       }
    }
}

int main(){
    int i,j;
    int x, y;
    int instancia=1;

    while( scanf("%d %d",&n,&m) > 0){

          memset(mat,AZUL,sizeof(mat));
          memset(cor,AZUL,sizeof(cor));
          memset(d,AZUL,sizeof(d));

          for(i=1; i<=m;i++){
               scanf("%d %d",&x,&y);
               mat[x][ d[x]++ ] = y;
               mat[y][ d[y]++ ] = x;
          }

          bipartido=1;
          Bipartido(1,PRETO);

          printf("Instancia %d\n",instancia++);
          if( bipartido )
              printf("sim\n\n");
          else
              printf("nao\n\n");

    }

    return 0;
}
```

25. Conectividade em grafos

```
#include <stdio.h>
#include <string.h>

#define BRANCO 0
#define PRETO 1

int mat[101][101];
int mark[101];
int d[101];

void DFS(int v){
    int i;
    int viz;
    mark[v]=PRETO;

    for(i=0; i<d[v]; i++){
         viz = mat[v][i];
         if( mark[viz] == BRANCO )
                DFS(viz);
    }
}

int main(){
    int i,j;
    int e, l;
    int x, y;
    int flag, teste=1;

    while(1){
          scanf("%d %d",&e,&l);
          if( (e+l)==0 )break;

          memset(mat,BRANCO,sizeof(mat));
          memset(mark,BRANCO,sizeof(mark));
          memset(d,BRANCO,sizeof(d));

          for(i=1; i<=l;i++){
               scanf("%d %d",&x,&y);
               mat[x][ d[x]++ ] = y;
               mat[y][ d[y]++ ] = x;
          }

          DFS(1);

          flag=1;
          for(i=1; i<=e;i++){
                 if( mark[i] == BRANCO ){
                     flag = 0;
                     break;
                 }
          }

          printf("Teste %d\n",teste++);
          if( flag )
              printf("normal\n\n");
          else
              printf("falha\n\n");
    }
    return 0;
}
```

26. Comparar duas potenciações

Para comparar duas potenciações (a^b, c^d) ambas com expoente muito grande sem causar overflow, vamos utilizar a função logarítmica, transformando multiplicações em adições.

$$a^b > c^d \;\rightarrow\; \ln(a^b) > \ln(c^d) \;\rightarrow\; b * \ln(a) > d * \ln(c)$$

```
#include <stdio.h>
#include <math.h>

int main(){
    int a=2,b=5; // 2^5=32
    int c=3,d=5; // 3^5=243

    if(  b*log(a) > d*log(c)  )
          printf("%d^%d > %d^%d\n",a,b,c,d);
    else
          printf("%d^%d > %d^%d\n",c,d,a,b);
    return 0;
}
```

27. Standard Template Library: Containers

27.1 Map

```
#include <map>

//chave, valor
map<char,int> Mapa;
map<char,int>::iterator it;

Mapa['a'] = 100;
Mapa['b'] = 200;
Mapa['c'] = 300;

/* limpar map */
// Mapa.clear();

/* tamanho do map*/
// Mapa.size();

/* Escontrar elemento */
//  map<char,int>::iterator i;
//  i = Mapa.find('a');

/* Remover um elemento */
// Mapa.erase( Mapa.find('b') );

/* Map está vazio? */
// Mapa.empty();

for(it=Mapa.begin(); it != Mapa.end(); it++){
   printf("%c=%d\n",(*it).first,(*it).second);
}
```

27.2 Vector

```
#include <vector>

vector<int> v;
vector<int>::iterator it;

for(int i=0; i<10; i++){
    v.push_back(i+1);
}

/* limpar vector */
// v.clear();

/* vector está vazio?*/
// v.empty();

/* erase the 6th element */
// v.erase (v.begin()+5);

/* erase the first 3 elements: */
// v.erase (v.begin(),v.begin()+3);

/* tamanho do vector */
// v.size();

/* Remover último elemento */
// v.pop_back();

for(int i=0; i<10; i++){
    printf("%d\n",v[i]);
}

for(it=v.begin(); it < v.end(); it++){
    printf("%d\n",(*it));
}
```

27.3 Stack

```
#include <stack>

stack<int> pilha;

for(int i=1; i<=10; i++){
     pilha.push(i);
}

printf("Size %d\n",pilha.size());

while(!pilha.empty()){
    printf("%d ",pilha.top());
    pilha.pop();
}
```

27.4 Queue

```
#include <queue>

queue<int> fila;

for(int i=1; i<=10; i++){
    fila.push(i);
}

printf("Size %d\n",fila.size());
printf("Last element %d\n",fila.back());

while(!fila.empty()){
      printf("%d ",fila.front());
      fila.pop();
}
```

27.5 Set

```
#include <set>

set<int> myset;
set<int>::iterator it;

for(int i=1; i<=5; i++){
    //10, 20, 30, 40, 50
    myset.insert(i*10);
}

printf("Size %d\n",myset.size());

// Se não encontrar retorna end()
it=myset.find(20);
myset.erase(it);
myset.erase( myset.find(40) );

for(it=myset.begin(); it!=myset.end(); it++){
     printf("%d\n",(*it));
}
// 10, 30, 50
```

27.6 List

```
#include <list>

list<int> mylist;
list<int>::iterator it;

/*
for(int i=1; i<=5; i++){
   mylist.push_back(i);
}

while(!mylist.empty()){
    printf("%d\n",mylist.front());
    mylist.pop_front();
}
//1,2,3,4,5
*/

/*
for(int i=1; i<=5; i++){
    mylist.push_front(i);
}

while(!mylist.empty()){
   printf("%d\n",mylist.back());
   mylist.pop_back();
}
*/

for(int i=1; i<=5; i++){
   mylist.push_back(i);
}
```

```
// inverte a orden da lista
mylist.reverse();


// remove toda ocorrência do
mylist.remove(5);

// ordena
mylist.sort();

//mylist.clear();
//mylist.size();
//mylist.empty();
//mylist.merge(list2);
//mylist.front();
//mylist.back();

for(it=mylist.begin(); it!=mylist.end(); it++){
     printf("%d\n",(*it));
}
```

28. Miscellaneous

28.1 Utility

```
pair <int,int> prod1 (1,2);
pair <int,int> prod2 = make_pair(23,4);
printf("%d %d\n",prod1.first, prod1.second);
printf("%d %d\n",prod2.first, prod2.second);
```

29. Algorithm

29.1 Max

```
printf("max(1,2)==%d\n",max(1,2));
printf("max(2,1)==%d\n",max(2,1));
printf("max('a','z')==%d\n",max('a','z'));
printf("max(3.1,2.7)==%.2f\n",max(3.1,2.7));
```

29.2 Min

```
printf("min(1,2)==%d\n",min(1,2));
printf("min(2,1)==%d\n",min(2,1));
printf("min('a','z')==%d\n",min('a','z'));
printf("min(3.1,2.7)==%.2f\n",min(3.1,2.7));
```

29.3 Sort

```
bool myfunction (int i,int j){
     return (i<j);
}

int myints[] = {32,71,12,45,26,80,53,33};
vector<int> myvector (myints, myints+8);

sort (myvector.begin(), myvector.end(),
myfunction);
```

29.4 Binary Search

```
int myints[] = {1,2,3,4,5,4,3,2,1};
vector<int> v(myints,myints+9);
// 1 2 3 4 5 4 3 2 1

// using default comparison:
sort (v.begin(), v.end());

if (binary_search (v.begin(), v.end(), 3))
    cout << "found!\n";
else
    cout << "not found.\n";
```

30. Modelo de código

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include <iostream>
#include <utility>
#include <algorithm>

#include <map>
#include <set>
#include <vector>
#include <list>
#include <queue>
#include <stack>

using namespace std;

#define abs(a) ((a) > 0 ? (a) : -(a))


int main(){
    int n;

    // fim da entrada EOF
    while ( scanf("%d",&n) > 0){
    }

    // fim de entrada 0
    while(1){
        scanf("%d",&n);
        if(n==0)break;

    }

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

    }

    system("pause");
    return 0;
}
```

31. Avaliação de chaves, colchetes e parênteses

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
#include <stack>

using namespace std;

int Valido(char a, char b){
     if( a+b == '('+')' ||
         a+b == '{'+'}' ||
         a+b == '['+']')
         return 1;
      else
         return 0;
}


int main(){
    int i;
    int t, flag;
    char ch;
    char v[100001];
    stack<char> mystack;


    scanf("%d",&t);

    while(t--){
    scanf("%s",v);

    flag=1;
    for(i=0; v[i]!='\0'; i++){
       if( v[i] == '(' ||
           v[i] == '{' ||
           v[i] == '[' )
              mystack.push(v[i]);
       else{
          if( !mystack.empty() ){
            ch = mystack.top();
            mystack.pop();

            if( !Valido(ch,v[i]) ){
                  flag=0;
                  break;
            }

          }else{
              flag=0;
              break;
          }
       }
    }

    // existe algo na pilha
    if( !mystack.empty() )
        flag=0;

    if(flag)
        printf("S\n");
    else
        printf("N\n");

     while(!mystack.empty())
         mystack.pop();
    }

    return 0;
}
```

32. String

```
#include <string>

string str1;
string::iterator it;

for(it=str.begin(); it<str.end(); it++)
   cout << *it;

/* entrada: se então*/
getline(cin,str1);
cout << str1;
/* saída: se então*/

/* entrada: se então*/
cin >>
cout << str1;
/* saída: se*/

str1.length();
str1.size();
str1.clear();
str1.empty();
str1.erase( str1.begin()+2 );
//possição da 1ª ocorrência
int n = str1.find("sol");

for (int i=0; i < str.length(); i++){
   cout << str[i];
}

string name ("John");
name+= "Carter";

string res, str2, str3;
str1 = "Test string: ";
str2 = 'x';
res = str1 + str2;
cout << res  << endl;
```

33. Faster algorithm for primality test [4]

```
int isPrime(long n){

    if(n==1)
        return 0;
    else if(n<4)
        return 1;
    else if(n%2==0)
        return 0;
    else if(n<9)
        return 1;
    else if(n%3==0)
        return 0;
```

```
    else{

        long r=(int)sqrt(n);
        int f=5;
        while(f<=r)
        {
            if(n%f==0) return 0;
            if(n%(f+2)==0) return 0;
            f=f+6;

        }
        return 1;
    }
}
```

34. Referências

1. http://www.ime.usp.br/~pf/analise_de_algoritmos/aulas/mochila-subsetsum.html
2. http://br.spoj.pl/problems/MOEDAS/
3. http://pt.wikipedia.org/wiki/Grafo_bipartido
4. http://www.bytehood.com/primality-test/5/