**<https://mattomatti.com/pl/a68>**

****

**Implementacja**

**Funkcja główna**

Przypuśćmy, że funkcja **zapakujZachlannie()** otrzymuje następujące argumenty: *p\_waga* - lista zawierająca wagi *n* elementów, *p\_wart* - lista zawierająca wartość *n* elementów, *n* - określa ile jest różnych przedmiotów oraz *U* - maksymalna waga jaką można spakować. Wtedy algorytm wygląda następująco:

1. double zapakujZachlannie(double\* p\_waga, double\* p\_wart, int n, double U) {
2. sortuj(p\_waga, p\_wart, n);
3. double W = 0;
4. int sztuk = 0;
5. for(int i = 0; i < n; i++){
6. sztuk = U / p\_waga[i];
7. U -= sztuk \* p\_waga[i];
8. W += sztuk \* p\_wart[i];
9. }
10. return W;
11. }

(1.) Wynikiem funkcji będzie łączna wartość zapakowanych przedmiotów. (2.) Dane na początku należy posortować względem opłacalności. Deklaracja (3.) zmiennej *W*, która będzie zliczać wartości przedmiotów oraz (4.) *sztuk* - zmienna, która przechowa w *i*-tej iteracji ile sztuk można zapakować. (5.) Dla każdego przedmiotu należy: (6.) wyliczyć ile sztuk można zapakować, a następnie symulując pakowanie: (7.) zmniejszyć wartość ile można jeszcze zapakować oraz (8.) zwiększyć wartość plecaka. Na koniec (10.) należy zwrócić *W*.

**Sortowanie danych**

Danych nie da rady posortować względem tylko jednej listy - trzeba tu napisać niestandardowe sortowanie, które uwzględni wartości z jednej jak i drugiej listy. Przykładowa implementacja na podstawie sortowania bąbelkowego wygląda następująco:

1. void sortuj(double\* p\_waga, double\* p\_wart, int n){
2. double t;
3. for(int i = 0; i < n - 1; i++){
4. for(int j = 0; j < n - i - 1; j++){
5. if(p\_wart[j]/p\_waga[j] <= p\_wart[j+1]/p\_waga[j+1]){
6. t = p\_waga[j];
7. p\_waga[j] = p\_waga[j + 1];
8. p\_waga[j + 1] = t;
9. t = p\_wart[j];
10. p\_wart[j] = p\_wart[j + 1];
11. p\_wart[j + 1] = t;
12. }
13. }
14. }
15. }

Różnica względem zwykłego sortowanie bąbelkowego polega na tym, że (1.) podane są dwie listy w argumentach i (5.) podczas porównywania jest wyliczana opłacalność każdego elementu. W przypadku, gdy elementy należy zamienić należy pamiętać, że trzeba tego dokonać na obu listach! Oczywiście do sortowanie można utworzyć dodatkową listę z wyliczonymi wartościami i użyć całkowicie innej metody sortowania.

**Testowanie funkcji**

W celu przetestowania programu można skorzystać z poniższej funkcji, która oczekuje kolejno: *n* - ile jest różnych przedmiotów, *U* maksymalny udźwig plecaka. Następne *n* elementów określa wartości elementów, a kolejne *n* ich wagę. Jako wynik podana jest łączna wartość przedmiotów plecaków zapakowana zgodnie z algorytmem zachłannym.

1. int main () {
2. int n;
3. double U;
4. cin >> n >> U;
5. double\* p\_wart = new double[n];
6. for(int i = 0; i < n; i++)
7. cin >> p\_wart[i];
8. double\* p\_waga = new double[n];
9. for(int i = 0; i < n; i++)
10. cin >> p\_waga[i];
11. cout << zapakujZachlannie(p\_waga, p\_wart, n, U);
12. cout << endl;
13. delete[] p\_waga, p\_wart;
14. system("pause");
15. return 0;
16. }

///-----------------------------------------

/\*

 http://mattomatti.com/pl/a68

\*/

#include <iostream>

using namespace std;

void sortuj(double\* p\_waga, double\* p\_wart, int\* p\_pos, int n){

 double t;

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

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

 if(p\_wart[j]/p\_waga[j] <= p\_wart[j+1]/p\_waga[j+1]){

 swap(p\_waga[j], p\_waga[j+1]);

 swap(p\_wart[j], p\_wart[j+1]);

 swap(p\_pos[j], p\_pos[j+1]);

 }

 }

 }

}

int\* zapakujZachlannie(double\* p\_waga, double\* p\_wart, int n, double U) {

 int\* p\_pos = new int[n];

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

 p\_pos[i] = i;

 sortuj(p\_waga, p\_wart, p\_pos, n);

 int\* p\_sztuk = new int[n];

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

 p\_sztuk[p\_pos[i]] = U / p\_waga[i];

 U -= p\_sztuk[p\_pos[i]] \* p\_waga[i];

 }

 delete[] p\_pos;

 return p\_sztuk;

}

int main () {

 int n;

 double U;

 cin >> n >> U;

 double\* p\_wart = new double[n];

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

 cin >> p\_wart[i];

 double\* p\_waga = new double[n];

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

 cin >> p\_waga[i];

 int\* p\_sztuk = zapakujZachlannie(p\_waga, p\_wart, n, U);

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

 cout << p\_sztuk[i] << " ";

 delete[] p\_waga, p\_wart, p\_sztuk;

 system("pause");

 return 0;

}