

Dynamiczne przydzielanie zasobów

Barbara Kosior



WYDZIAŁ MATEMATYKI
i INFORMATYKI
Uniwersytet Łódzki

Czym jest zasób ?

Określenie **zasoby systemowe** używane jest to określania ilości pamięci RAM, jaką komputer dysponuje. W pamięci tej przechowywane są aktualnie wykonywane programy oraz dane dla tych programów i wyniki ich pracy. To z tej pamięci korzystają programy, które tworzymy.

Co to alokacja pamięci ?

Alokacja pamięci odnosi się do metod rezerwowania pamięci RAM wykorzystywanej do realizacji programów komputerowych. Określa się ją jako przydział obszaru pamięci który system przydziela z puli wolnej przestrzeni. Możemy wyróżnić 2 sposoby alokacji : statyczną oraz dynamiczną. W zależności jaki typ pamięci (statyczny bądź dynamiczny) zostanie wybrany program może przydzielić pamięć na początku działania programu lub robić to w trakcie wykonywania.

Przeciwieństwem alokacji jest dealokacja pamięci. Określa się ją jako zwolnienie ciągłego obszaru pamięci.

Pamięć statyczna a dynamiczna?

STATYCZNA

Pamięć zostaje przydzielona przed rozpoczęciem wykonywania programu i istnieje do momentu jego zakończenia

Pamięć jest zwalniana po zakończeniu programu.

Z wyprzedzeniem należy przewidzieć wielkość obszaru jaki będzie potrzebny.

Wykorzystywany jest stos.

Najbezpieczniejsza metoda.

DYNAMICZNA

Rozmiar nie może być ustalony (przewidziany) w momencie implementacji programu.

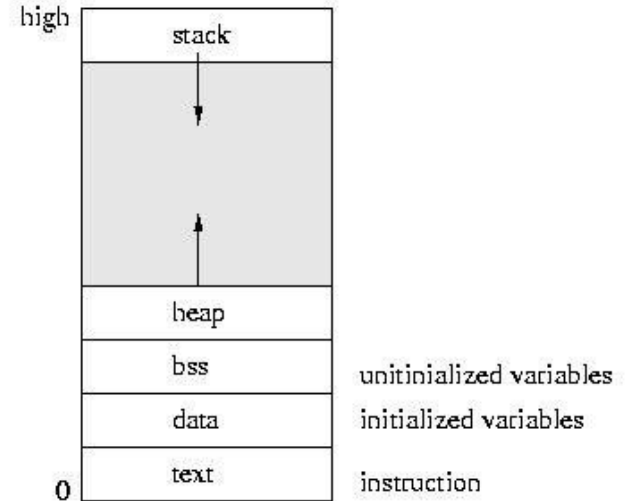
Za zwolnienie pamięci odpowiada programista.

Pamięć jest przydzielana ze sterty i jest jej przydzielane tyle ile program potrzebuje w danej chwili.

Realizowana jest jako sterta.

Krótko co to sterta ...

Sterna to obszar pamięci którego części przydzielane są na wyłączność uruchomionym programom (procesom). Nie jest jednak to tak jak stos pamięć uporządkowana, i nie następuje jej samoistne czyszczenie po zakończeniu funkcji. Pamięcią ze sterty może zarządzać programista, może on na stercie umieszczać dane, nadpisywać, odczytywać, usuwać itd.. Rozmiar tego obszaru dla aplikacji może być bardzo duży, a pamięć jest dynamicznie rozszerzana gdy zachodzi taka potrzeba. Miejsce na stercie możemy rezerwować oraz zwalniać w dowolnym momencie działania programu. Jednak by nie doszło do przepełnienia się pamięci powinno się zwalniać pamięć gdy dane w niej umieszczone nie są już potrzebne.



Podczas korzystania z pamięci przydzielanej dynamicznie możliwe jest skierowanie prośby w dowolnym momencie działania programu o przydzielenie nowych obszarów pamięci. System zarządzając stertą wykorzystuje tabelę, w której zapisywany jest adres początkowy i rozmiar w bajtach każdego przydzielonego bloku pamięci.

Sekwencja korzystania z dynamicznej alokacji pamięci

Krok	Operacja	Uwaga
Wniosek o przydział w ramach programu	Program prosi system operacyjny o przydzielenie miejsca w pamięci [poprzez określenie żądanej liczby bajtów].	
Alokacja według systemu operacyjnego	Jeśli blok bajtów [pamięci] o żądanym rozmiarze jest dostępny w stercie, system operacyjny zwraca adres początkowy tego bloku dostępnych bajtów.	<ul style="list-style-type: none"> • ten blok bajtów jest następnie rezerwowany dla programu (mówimy, że jest do niego przydzielony); • dlatego ten blok bajtów nie jest już dostępny dla żadnego innego programu.
Wykorzystanie przydzielonej przestrzeni pamięci	Program wykorzystuje przydzieloną przestrzeń pamięci według własnego uznania.	
Żądanie zwolnienia przez program	Gdy program nie potrzebuje już przydzielonej przestrzeni pamięci, zwalnia ją, wskazując systemowi operacyjnemu adres początku zarezerwowanego bloku, który chce zwolnić.	Jeśli program nie zwalnia przydzielonej pamięci przed wyjściem, powoduje to wyciek pamięci .

Cechy zmiennych dynamicznych

- ★ Zmienna nie ma nazwy, dostępna jest poprzez wskaźnik
- ★ Istnieje ona od momentu alokacji do zwolnienia pamięci
- ★ Zmienna jest wszędzie dostępna, o ile dostępny jest wskaźnik do niej
- ★ Zmienna może zawierać “śmieci”
- ★ Przydział pamięci odbywa się z tzw. sterty

Alokacja dynamiczna pamięci

W przypadku języka C++ do alokacji pamięci wykorzystuje się operator **new**. Po operatorze podajemy jaki typ mają mieć przechowywane dane. Operator new zwraca wskaźnik do miejsca pamięci w której została dokonana alokacja.

type *name = new type

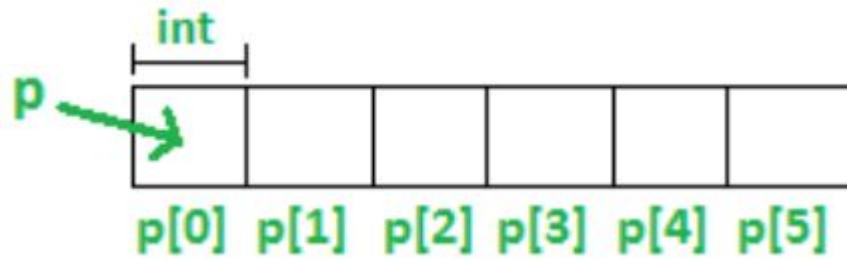


typ deklarowanej zmiennej

wskaźnik do pamięci

przydzielenie pamięci i zwrócenie wskaźnika

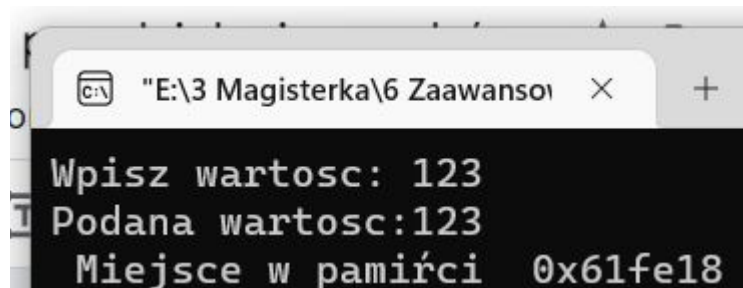
int *p = new int [5] - > przykład alokacji pamięci z wartością początkową równą 5



Jak sprawdzić czy alokacja się udała?

Przydział pamięci nastąpił gdy operator new zwraca wskaźnik inny niż zerowy.

```
oper_new.cpp
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      int *wskaznik = new int;
7      // sprawdzenie czy przydzielono pamięć
8      if (wskaznik!=NULL)
9      {
10         //pamięć przydzielona
11         cout << "Wpisz wartosc: ";
12         cin>> *wskaznik;
13         cout << "Podana wartosc:" << wskaznik[0] << " \n";
14         cout << " Miejsce w pamieci  " << &wskaznik;
15         //zwolnienie pamieci
16         delete wskaznik;
17     }
18     else
19     {
20         // nie udało się przydzielić pamięci
21         cout << "Brak pamieci";
22     }
23 }
24 }
```



```
"E:\3 Magisterka\6 Zaawanso"
Wpisz wartosc: 123
Podana wartosc:123
Miejsce w pamieci 0x61fe18
```

Wskaźniki

Wskaźnik to zmienna, która zawiera adres pierwszej komórki pamięci, w której przechowywane jest inna zmienna. Gdy zmienna zajmuje więcej niż jedną komórkę pamięci to wskaźnik wskazuje na pierwszą z tych komórek.

Operatory :

& - operator pobrania adresu , zwraca numer komórki pamięci zmiennej, która jest jego argumentem

***** - operator wyłuskania , pozwala na odwołanie się do zawartości komórki pamięci

p = &a; do zmiennej p przypisana zostaje liczba będąca adresem zmiennej a

***p += 7;** do zmiennej umieszczonej pod adresem p dodawane jest 7

Wskaźniki

Wskaźnik danego typu powinien być wykorzystywany do przechowywania adresów zmiennych tego samego typu.

```
char *wsk_c; // wskaźnik do zmiennej typu char  
char int *wsk_i; // wskaźnik do zmiennej typu int  
long *wsk_l; // wskaźnik do zmiennej typu long  
float *wsk_f; // wskaźnik do zmiennej typu float  
double *wsk_d; // wskaźnik do zmiennej typu double
```

W przypadku gdy przydział pamięci się nie powiedzie operator **new** zwróci wskaźnik zerowy (NULL). Wydarzyć się może tak w przypadku gdy chcemy zarezerwować zbyt duży blok pamięci bądź system nie posiada już wolnych zasobów.

Przykład

```
1
2  #include <iostream>
3  #include <string>
4  using namespace std;
5
6  int main() {
7      //zadeklarowanie zmiennej typu double
8      double *pd = new double(4.5);
9      //wypisanie
10     cout << "*pd = " << *pd << endl;
11     // zmiana wartości
12     *pd = 10.5;
13     // ponowne wypisanie po zmianie
14     cout << "*pd = " << *pd << endl;
15     // zwolnieniej pamięci
16     delete pd;
17 }
18
```

Efekt:



```
"E:\3 Magisterka\6 Zaawanso... × +
*pd = 4.5
*pd = 10.5
```

Zwalnianie pamięci przydzielonej dynamicznie

Do zwalniania pamięci przydzielonej dynamicznie służy operator **delete** (w języku C++). Zwolnienie pamięci pozwala na zwrócenie zwolnionej pamięci do puli pamięci dostępnej do ponownego wykorzystania.

Jeśli pamięć dla danych, na które wskazuje zmienna została przydzielona bez podania parametru określającego ilość elementów wykorzystujemy składnię :

delete zmienna;

Jeśli przydzielono pamięć z użyciem parametru określającym ilość elementów to musimy poinformować wskaźnik o tym że wskazano tablicę rekordów. Za operatorem delete dopisujemy nawias kwadratowy. Operator ustala rozmiar bloku jaki został przydzielony i go usunie.

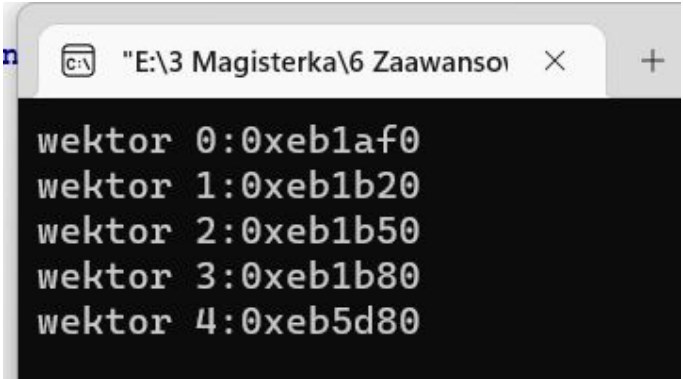
delete[] wskaznik_do_tablicy;

Zwalnianie pamięci przydzielonej dynamicznie - cd.

Ważne przy zwalnianiu pamięci przez operator **delete** jest unikanie zwalniania tego samego obszaru pamięci więcej niż jeden raz. Może to później skutkować problemami z wykonywaniem programu. Najlepszym rozwiązaniem jest zwolnienie pamięci przed użyciem wskaźnika oraz po każdym zwolnieniu wskazywanego przez niego obszaru, gdy ten obszar nie jest nam już potrzebny. W korzystaniu z języka C++ sami musimy zadbać o zwolnienie wykorzystywanej pamięci ale istnieją języki wysokiego poziomu które oferują mechanizmy odśmiecania pamięci (np. Java, PHP, Python).

Przykład

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      //tablica 2 wymiarowa
7      int **wektory = new int *[5];
8      for (int i = 0; i < 5; ++i)
9      {
10         wektory[i] = new int [10];
11         cout<< "wektor " << i << ":" <<wektory[i]<<"\n";
12     }
13     delete [] wektory ;
14 }
15
16
```



```
wektor 0:0xeb1af0
wektor 1:0xeb1b20
wektor 2:0xeb1b50
wektor 3:0xeb1b80
wektor 4:0xeb5d80
```

Wyciek pamięci - co to takiego ?

Do wycieku pamięci dochodzi w momencie zapomnienia o wywołaniu **delete** na koniec działania programu. Powoduje to że pamięć nie jest zwracana do ponownego wykorzystania przez system operacyjny. Z każdym następnym wywołaniem **new** system przydziela nową pamięć. Jeśli program się zakończył bądź zgubiono wskaźnik do jego lokalizacji to pamięć cały czas jest zajęta. Może to spowodować zawieszenie się programu lub całego systemu gdy skończy się dostępna pamięć.

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int main() {
6      while (true) {
7          int *i = new int;
8      }
9      return 0;
10 }
11
12
```

Przykład jak może wyglądać program, który spowoduje wyciek pamięci.

Referencja

Jest to wartość, która zawiera informacje o położeniu w pamięci innej wartości. Referencja przypomina w swoim działaniu wskaźniki, które zostały wcześniej omówione. Jednak do referencji można przypisać adres tylko raz. Referencja zawiera w sobie adres i typ tej zmiennej ale nie wymaga stosowania operatora * (jako wskaźnik). W referencji wygodniejszy jest dostęp do zmiennej.

typ &NazwaReferencji = wartość;

Od tej pory można używać obu tych zmiennych zamiennie.

```
void main()
{
    int a=0, b=0;
    int &ref = a; -> ref ma wart 0
    ref = 10;      -> ref ma wart 10 , a=10, b=0
    ref = b;       -> ref ma wart 0 , a=0, b=0
    ref = 20;      -> ref ma wart 20 , a=20, b=0
}
```



WYDZIAŁ MATEMATYKI
i INFORMATYKI
Uniwersytet Łódzki

Dziękuję za uwagę