Элементы ЯПВУ. УКАЗАТЕЛИ.

Pascal

C / C++

Динамические переменные

Вся динамическая память в Pascal это сплошной массив байтов (куча).
Адрес начала кучи храниться в переменной HeapOrg, адрес конца в HeapEnd, а текущую нижнюю границу свободной динамической памяти адресует указатель HeapPtr.

Создаваемые в этой памяти динамические переменные не имеют собственных имен – к ним обращаются через указатели.

При завершении программы используемая её динамическая память автоматически освобождается.

Указатели в C\C++ используются при создании и обработке динамических объектов. Память для них называется динамически распределяемой областью памяти, heap, кучей.

Динамические объекты, в отличие от заранее определяемых, создаются и уничтожаются динамически и явно в процессе выполнения программы. Динамические объекты не имеют имен, и ссылка на них выполняется с помощью указателей.

При завершении программы используемая её динамическая память автоматически освобождается.

Элементы ЯПВУ. УКАЗАТЕЛИ.

Динамические переменные

Pascal

Для ***выделения памяти*** типизированному указателю служит стандартная процедура new():

- **New (P);** где P – имя указателя;

[new можно исп. и в виде функции]

Но для указателя типа Pointer исп.:

- **Getmem (P; Size);**

где Size - к-во байт памяти.

За одно обращение к куче getmem может зарезервировать 65 521 байт памяти.

Для ***освобождения памяти***, выделенной процедурой (функцией) new(), исп.:

- **Dispose (P);**

От getmem освобождает память

- **Fremem (P; Size);**

После освобождения памяти значение указателя становится *неопределенным*, поэтому во избежание проблем его лучше сразу же "обнулить":

```
dispose(p);
p:= nil;
```

C

Для ***выделения памяти*** указателю в С используются функции из stdlib.h:

- **malloc (size);** - возвращает указатель на выделенный блок памяти размером size байт.
- **calloc (num, size);** - возвращает указа-тель на первый байт выделенного блока памяти размером num элементов каждый по size байт (все элементы инициализируются нулями).

Если память не выделена значение возвращаемого указателя NULL.

Для ***изменения выделенной памяти*** исп.:

- **realloc (prt, size);** - получает блок памяти (указатель prt), изменяет его размер до size (больший или меньший прежнего) и возвращает указатель на новое положение блока памяти (другое, если блок пришлось переместить). Ранее размещенные данные сохраняются. Если не удается увеличить блок, то возвращается NULL, а значение prt остается доступным.

Для ***освобождения памяти*** используется:

- **free (prt);** - где prt – имя указателя, адресующего блок памяти;

Элементы ЯПВУ. УКАЗАТЕЛИ.

Pascal | **Динамические переменные** | **C / C++**

Для ***освобождения памяти*** одновременно из-под нескольких переменных применяют процедуры:

- **Mark (P);**

она вызывается до начала выделения памяти и записывает в указатель P адрес начала динамической памяти (из переменной HeapOrg). Затем можно процедурой

- **Release (P);**

освободить участок памяти занятый до момента вызова *release*, начиная с адреса записанного в *mark*.

Вспомогательные функции:

- **Maxavail** – длина в байтах самого длинного свободного участка памяти (тип функции longint),
- **Memavail** – полный объем свободной динамической памяти в байтах (тип функции longint),
- **Sizeof (X)** – объем в байтах занимаемый X, где X – либо имя переменной любого типа, либо имя типа (тип функции word).

Примеры:

- int* n = (int*) malloc (sizeof(int));
- float *p; p = calloc (100, sizeof(float));
- char* p; p = realloc (p, 18);

Для ***выделения памяти*** указателю в C++ используются функции:

- **new <тип_данных>;** - возвращает указатель на первый байт блока памяти достаточного для размещения данных указанного типа.

Если память выделить не удалось – обрабатывается исключительная ситуация bad_alloc (в старых компиляторах – возвращается 0).

Для ***освобождения памяти*** используется:

- **delete prt;** - где prt – имя указателя, адресующего освобождаемый блок памяти.

Примеры:

- int* n = new int;
- int* o = new int (12); - начальное значение
- delete n; delete o;
- int* kl = new int [12]; - указатель на массив
- delete [] kl; (без [] освободит первый элемент массива, а остальные станут недоступны, но заняты - мусор).

Элементы ЯПВУ. УКАЗАТЕЛИ.

Pascal

Указатели на процедуры и функции

Указатели на подпрограмму являются переменной процедурного (функционального) типа:

Имя функции (процедуры) – это константа процедурного типа, поэтому присваивание происходит без операции взятия адреса @ , т.к. в этом типе имя подпрограммы - и есть адрес точки входа в неё.

```
type   fun =  function (x : real) : real;
  var pf : fun;
(* конкретная функция *)
function f (x : real) : real : far;
Begin
  ……                    (* тело функции *)
End;
  pf := f;
```

В переменной pf теперь содержится адрес точки входа в функцию f, и можно вызывать её так: `y := pf(x);`

Функция должна компилироваться в режиме дальней адресации - far, ведь адрес должен содержать сегмент и смещение, а без far – одно смещение, т.к. компилятор формирует всего один сегмент кода.

C / C++

Указатель на массив

Указатель на массив хранит адрес его первого байта.

Имя массива без индексов – это его адрес.

Пример:

```
char str[50],  *pl;
pl = str;
```

str[4] и *(pl+4) - эквивалентны

Массивы указателей

Как и объекты любых других типов, указатели могут быть собраны в массив.

```
int *x[20];
```

x[2] = &va – присвоение адреса переменной va третьему элементу массива указателей x.

Указатель на функции

Указатель на функцию содержит адрес в сегменте кода, по которому располагается исполняемый код функции. Имя функции без скобок и аргументов – это адрес точки входа в неё. Вид указателя на функцию:

```
<тип>  (*<имя>)  (<список_типов_
                        аргументов>)
```

Пример:

```
int (*p) (const char *, const char);
```

Элементы ПЯВУ. УКАЗАТЕЛИ.

Pascal

Определить, что делает программа:

C / C++

```
Program memo;
Type
  mas_int = array [1..maxint]  of  integer;
Var
  p   : ^mas_int;
  i, n : integer;
Begin
 Writeln ('Введите размер массива:');
 Readln (n);
 If Maxavail < n * Sizeof (integer) then
   Begin
      writeln(' Недостаточно памяти');
        halt;  (*немедленно завершает выполнение программы *)
    End;
 Getmem (p, n * Sizeof (integer));
 for I := 1 to n do
   read (p^[i]);
End.
```

Эта программа вводит массив целых чисел, размер которого неизвестен на стадии компиляции, а запрашивается во время выполнения программы. Перед выделением памяти проверяется наличие свободного места.

```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
int main()
{
  char s[] = "2, 38.5, 70, 0, 0, 1,", *p = s;
  double m[10];
  int i = 0;
do
   {
 m[i++] = atof(p);
  if (i>9) break;
   }
while (p = strchr(p, ','), p++);
for (int k=0; k<i; k++)
  printf ("%5.2f ", m[k]);
return 0;
}
```

Программа заполняет массив типа double из строки s