Элементы ЯПВУ. УКАЗАТЕЛИ.

УКАЗАТЕЛЬ – это переменная, значением которой является адрес какого-то объекта в памяти компьютера (обычно, другой переменной).

Для каждой описанной в программе переменной выделяется область памяти, когда мы объявляем в программе, например, float der, то выделяется постоянная (статическая) область памяти, которая будет занята этой переменной до конца работы программы. Указатели, в частности, позволяют организовать динамическое выделение и освобождение памяти для переменных в процессе выполнения программы.

Но указатели используют и для работы с переменными в статической памяти.

С помощью указателей например можно:

- **обрабатывать многомерные и одномерные массивы, строки, символы, структуры и массивы структур , в том числе, большого размера,**
- **динамически создавать новые переменные и структуры в процессе выполнения программы,**
- **обрабатывать связанные структуры: стеки, очереди, списки, деревья, сети,**
- **передавать функциям адреса фактических параметров,**
- **передавать функциям адреса функций в качестве параметров.**

Разберемся с организацией памяти в ПЭВМ под MS DOS и в программах на С.

Адресация и распределение памяти для программ

ИНТЕРНЕТ

Для IBM совместимых ПЭВМ

Оперативная память (ОП) ПЭВМ состоит из ячеек размером в 1 байт, каждая ячейка имеет собственный уникальный номер – адрес. Память разделяется на сегменты размером 2^{16} = 65 536 байт или 64 Кб, что определяется архитектурой микропроцессоров 80x86 (16-разрядных). [32-х и 64-х-разрядная адресация поддерживается в операционных системах Windows и UNIX / Linux.]

Адрес каждого байта ОП формируется из двух слов (каждое размером по 16 бит – по 2 байта) – адреса сегмента и смещения (смещение указывает на сколько байт от начала сегмента сдвинут адрес конкретного байта).

Адрес байта формируется следующим образом: адрес сегмента смещается на 4-е двоичных разряда (бита) влево и к нему прибавляется смещение, таком образом физический адрес байта составляет 20 бит и с его помощью можно адресовать 2^{20} байт (1 Мбайт):

Указатель занимает в памяти 4 байта (32 бита) и адресует первый байт данных, например, первый байт первого элемента массива.

Указатель занимает в памяти 4 байта (32 бита) хотя собственная длина указателя – 20 бит и он помещается в 3 байта, НО обычно память выделяется словами по 2 байта (16 бит) и выравнивается на границу слова – т.е под указатель выделяется 2 слова (а не 1,5).

Адресация и распределение памяти для программ С

C / C++

Для IBM совместимых ПЭВМ и MS DOS

Для кода программы, данных и переменных, тип которых описан в программе, компилятор выделяет в оперативной памяти (ОП) DOS фиксированную память, не изменяющуюся в процессе выполнения программы – статическую память. Если невозможно заранее определить объем данных, то надо перейти к использованию динамической памяти. Её выделение под данные и освобождение осуществляется программистом. Расположение динамической памяти (heap, кучи) в компьютере показано на схемах ниже. Обращение к динамическим переменным осуществляется через указатели.

Карта оперативной памяти программ на C/C++ в MS DOS

Старшие адреса	Системная область DOS
SSeg:0000	Стек (растёт вниз)
	Динамическая память (heap, куча) (растёт вверх)
DSeg:0000	Глобальные переменные
CSeg:0000	Код программы
PrefixSeg	Префикс сегмента программы (PSP)
Младшие адреса	Системная область DOS

Адресация и распределение памяти для программ

Замечания к карте оперативной памяти для программ на С в MS DOS

Откомпилированная С-программа создает и использует 4 логически разделенных области памяти, имеющих свое назначение.

Первая область – это память, содержащая **код программы**. Код главной программы размещается в **кодовом сегменте (Cseg)**. Размер кодового сегмента не может превышать 64К.

Следующая область предназначена для **хранения глобальных переменных**. **Сегмент данных (адресуемый через DSeg)** содержит все глобальные переменные. Регистр DS никогда не изменяется во время выполнения программы. Размер сегмента данных не может превышать 64К.

Область **Стека** используется для самых различных целей при выполнении программы. Он содержит адреса возвратов вызываемых функций, аргументы, передаваемые в функции, и локальные переменные. Он также используется для хранения текущего состояния процессора. При запуске программы регистр **сегмента стека (SSeg)** указывает на первый байт после сегмента стекаРазмер стекового сегмента не может превышать 64К; размер по умолчанию - 16К, он может быть изменен директивой компилятора.

Область **Кучи** – это область свободной памяти, которую программа может использовать для динамического выделения памяти.

Префикс сегмента программы (Program Segment Prefix - PSP) -это 256-ти байтовая область, создаваемая DOS при загрузке программы.

Элементы ЯПВУ. УКАЗАТЕЛИ.

C / C++

УКАЗАТЕЛЬ – это переменная, значением которой является адрес какого-то объекта в памяти компьютера (обычно, другой переменной).

Указатель в С не является отдельным типом данных, он всегда связан с каким-либо конкретным типом.
В С / С++ различают три вида указателей:

на объект, на функцию и на void.

Объявление указателя на объект:

<тип> * <имя>;

Например: int *a, *c;

Объявление указателя на void – применяется когда конкретный тип объекта, адрес которого требуется хранить, не определен:

Например: extern void *malloc(int);

Объявление указателя на функцию:

<тип> (*<имя>) (<список_типов_аргументов>);

Например: int (*fun) (double, double);

Примеры:

```
int *pi;                    // Указатель на целую переменную
const int *pci;             // Указатель на целую константу
int * const cp=&i;          // Указатель-константа на переменную
const int *const cpc=&ci;   // Указатель-константа на целую константу
```

Элементы ЯПВУ. УКАЗАТЕЛИ.

C / C++

Операции с указателями

- **Получение адреса – &<имя_переменной> – значение адреса операнда <имя> в памяти.** Например: n = &p1;
- **Разыменование (разадресация, раскрытие ссылки) – *<имя_указателя> – значение объекта, адрес которого <имя_указателя>.**
  Например: z = *n;
- **Присваивание –**
  <имя_указателя1> = <имя_указателя2> -
  если оба операнда имеют один тип – простое присваивание, иначе – присваивание с преобразованием типа указателя – это:
- **Приведение типов – два вида – с указателем типа void* и без его использования.**
  void* в C разрешает неявное преобразование (в C++ – только явное, даже для void*). Преобразования всех остальных типов указателей должны быть явными, т.е. должна быть указана операция приведения типов.

Например: если x – имеет тип int, а *p – double, то надо записывать p = (double *) &x.

Использовать приведение надо осторожно и обязательно к базовому типу указателя, а не объекта.

Элементы ЯПВУ. УКАЗАТЕЛИ.

C / C++

Операции с указателями

- **Арифметические операции (адресная арифметика)** – суммирование и вычитание – учитывают размер типа величины и применимы только к указателем одного типа:
  - **сложение (вычитание) с константой** – p1 = p1 + 12; p2 = p2 – 5;
  - **увеличение (инкремент)** – p1++; – при увеличении на 1 указатель p1 будет смещаться на величину типа данных и указывать на следующее значение.

    Например:

    в типе int* – смещение на 2 байта – p1=200, p1++ = 202, p1-- = 198.
  - **уменьшение (декремент)** – p1--; – аналогично инкременту.
  - **вычитание двух указателей** – разность двух указателей – это разность их значений, деленная на размер типа в байтах. Так можно определять количество объектов (например, элементов массива) между адресами указателей.

    Например:

    в массиве разность указателей на 3-й и 6-й элементы равна 3.

- **Сравнение** - допустимы любые операции сравнения, обычно, они имеют смысл, когда ссылки на общий объект.

Элементы ЯПВУ. УКАЗАТЕЛИ.

C / C++

ПРИМЕР

В языке C/C++ с указателями допустимы практически все операции.

```
#include <iostream.h>
void main ()
{
int *a, b;
*a=100; b=200'
cout<<"\na="<<a<<", *a="*a<<", b="<<b;
}
```

Результат: a=0x034c, *a=100, b=200

Элементы ЯПВУ. УКАЗАТЕЛИ.

Динамические переменные

C / C++

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.

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

- **malloc (size); – возвращает указатель на выделенный блок памяти размером size байт.**
- **calloc (num, size); – возвращает указатель 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.**

Примеры:

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

Элементы ЯПВУ. УКАЗАТЕЛИ.

Динамические переменные

C / C++

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).

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.

Пример: int *x[20];

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

Указатель на функцию

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.

Определить, что делает программа:

C / C++

```
#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: