Указатели. Что это за штуки и на что они указывают? Давайте разбираться.

И так указатель – это переменная, значением которой является адрес ячейки памяти. О как , интересно. Адрес, ячейка памяти – это как вообще?   
  
Ну , для начала нужно немного вступительной теории. У вашего компьютера есть память! Память компьютера довольно сложная штука : Оперативная память, специальная кэш-память, регистры, Постоянная память и тд. и тп. Чтобы не лезть в эти дебри будем упрощать. Я предлагаю такой вид представления памяти:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Будем считать что каждая программа имеет свою личную табличку как показано выше. Каждая ячейка таблицы будет равна 1 байту.  
  
Давайте напишем код и посмотрим что произойдет с нашей памятью  
  
#include "pch.h"

#include <iostream>

using namespace std;

int main()

{

int a = 9;

}  
  
Вот так будет выглядеть наша память для этого кода:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  | a 9 | | | |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Опа,что-то произошло, давайте разбираться.Вопрос первый : сколько памяти занимает переменная типа int? Уверен что сразу же сказал – 4 байта. Присмотрись к таблице , 4 ячейки объединились и окрасились в синий , а внутри этой ячейки лежит 9 , а называется эта ячейка «а».  
  
Давай приведем ещё пример для наглядности:

#include "pch.h"

#include <iostream>

using namespace std;

int main()

{

int a = 9,b;

char c = 't', text;

double number = 34.212;

}

Вот так будет выглядеть наша память для этого кода:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Number 34.212 | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | b | | | |  |
| c ‘t’ | a 9 | | | |  |  |  |  |  |
|  |  |  |  |  |  | text |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

Красиво выглядит. Давай вместе с тобой проговорим что здесь произошло. Мы инициализировали переменную int и присвоили ей значение 9. В памяти из-за этого появилась ячейка на 4 байта с названием «а» и хранящая в себе число 9.Далее мы инициализировали переменную b , но не присвоили ей значение. В памяти из-за этого появилась ячейка на 4 байта с названием «b» и ничего в себе не хранящая (На самом деле В отличие от других языков программирования, C и C++ не инициализируют переменные определёнными значениями (например, нулём) по умолчанию. Поэтому при создании переменной ей присваивается ячейка в памяти, в которой уже может находиться какой-нибудь мусор! Но мы будем читать что там ничего нет). Далее мы инициализировали переменную «с» и присвоили ей значение «t». . В памяти из-за этого появилась ячейка на 1 байта с названием «с» и хранящая в себе символ «t». ). Далее мы инициализировали переменную «text» и присвоили ей значение «t». . В памяти из-за этого появилась ячейка на 1 байта с названием «text» которая ничего в себе не хранит. Далее мы инициализировали переменную «number» и присвоили ей значение «34.212». В памяти из-за этого появилась ячейка на 8 байта с названием «Number» которая хранит в себе число «34.212».

Ну говорить простыми словами то программа когда мы говорим ей создать переменную просто застолбила участок нужной длинны для её хранения в памяти.  
  
Теперь вернемся к указателям. Как вы уже наверное догадались указатель будет указывать на какой-то участок который компьютер застолбил для своей переменной. Теперь нужно понять как он это делает? Давайте вспомним определение указателя : указатель – это переменная, значением которой является адрес ячейки памяти. И так , мы сразу видим что указатель – это переменная , а раз это переменная , значит компьютер тоже должен застолбить для неё участок нужной длинны. Рассмотрим код:  
  
#include "pch.h"

#include <iostream>

using namespace std;

int main()

{

int a = 9,b;

char c = 't', text;

double number = 34.212;

double \*num;

char \*word;

}

Сколько памяти мы должны выделить для double \*num ? А сколько памяти для char \*word ? Мы не знаем. Давайте подумаем , а как вообще определяется сколько байт нужно какой переменной и почему? Почему для char хватает 1 байта , а int нужно 4? Все просто! В переменную char можно записать любое число от 0 до 255. Как вы помните в 1 байте 8 бит. Бит хранит в себе либо 0 , либо 1 и с помощью комбинирования 0 и 1 можно получить любое число в диапазоне от 0 до 255(Учите Булевскую алгебру,студенты). Для int было не достаточно максимального числа 255 , поэтому ему дали в 4 раза больше байт , только его диапазон увеличился не в 4 раза , а в гораздо больше −2 147 483 648 до 2 147 483 647 (Удивительная штука – эта двоичная система). Ну так главный вопрос:сколько нужно памяти указателю? Указателю нужно 8 байт и при этом не важно на какой тип данных он указывает,для double \*num и для char \*word понадобится 8 байт каждому.

Эмм,а не зажрался ли этот указатель!? 8 байт ему подавай! Что он вообще там хранит такое что ему 8 байт целых нужно?!

Вернёмся к определению указателя: указатель – это переменная, значением которой является адрес ячейки памяти. Значением которой является адрес ячейки памяти. Хмм. А как выглядит этот адрес ячейки памяти? Это хороший вопрос! Давай покажу.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A |
| B | C | D | E | F | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E |

Знакомьтесь – это 16-ая система! С помощью неё нумеруются байты в памяти.У десятого байта будет адрес A , у 3-го байта адрес 3 и тд. 16-ричная система используется для сокращенной записи. Байтов в памяти на много больше чем может вместить int , именно поэтому указатель такой большой и хранит он в себе 16-ричное число , адрес байта!

Теперь когда мы знаем сколько памяти нужно нашему указателю давайте посмотрим как все в памяти будет выглядеть для данного кода:

#include "pch.h"

#include <iostream>

using namespace std;

int main()

{

int a = 9,b;

char c = 't', text;

double number = 34.212;

double\*num;

char \*word;

}

Вот так будет выглядеть наша память для этого кода:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Number 34.212 | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | b | | | |  |
| c ‘t’ | a 9 | | | |  |  |  |  |  |
|  |  |  |  |  |  | text |  |  |  |
| Num | | | | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | word | | | | | | | |

Как мы видим, указатели заняли свою память и готовы записать в себя значения. Давайте запишем их раз указатели так этого хотят.

#include "pch.h"

#include <iostream>

using namespace std;

int main()

{

int a = 9,b;

char c = 't', text;

double number = 34.212;

double \*num = &number;

char \*word = &c;

}

Так , что эта за штучка «&»? Автор – поясни за код.  
  
Это унарная операция взятия адреса , она возвращает адрес 1-ого байта участка к которому мы этот оператор применили. Помните таблицу с адресами байтов?

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A |
| B | C | D | E | F | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 17 | 18 | 19 | 1A | 1B | 1C | 1D | 1E |

Что вернет &number? &number вернут 2 , так как наша переменная Number занимает байты с 2 по 9 ячейки. А что вернет &c? &c вернет 1F , так как char и так занимает только 1 ячейку.

Значит таблица памяти будет теперь выглядеть так:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Number 34.212 | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  | b | | | |  |
| c ‘t’ | a 9 | | | |  |  |  |  |  |
|  |  |  |  |  |  | text |  |  |  |
| Num 2 | | | | | | | |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  | Word 1F | | | | | | | |

Как вы видите Num хранит в себе адрес 2 , а не значение которое хранится по этому адресу 34.212.Это довольно частая ошибка у студентов. Не будь студентом – будь true программистом!

Теперь давайте разберемся а как выводить эти указатели?

Напишем код:

#include "pch.h"

#include <iostream>

using namespace std;

int main()

{

int a = 9, b;

double number = 34.212;

double \*num = &number;

cout << num << endl << \*num<<endl;

}

Чем отличается num от \*num спросишь ты меня? И отвечу я тебе : если написать cout << num , то тебе выведет адрес который хранится в этом указатели,тоесть адрес по которому располагается значение лежащее в number. А если ты напишешь cout << \*num , то тогда программа поймет что ты хочешь перейти по адресу который хранится в num и вывести его содержимое, в данном случае 34.212.Запусти этот код для более лучшего усвоения.

Введение по указателям на этом думаю закончу.